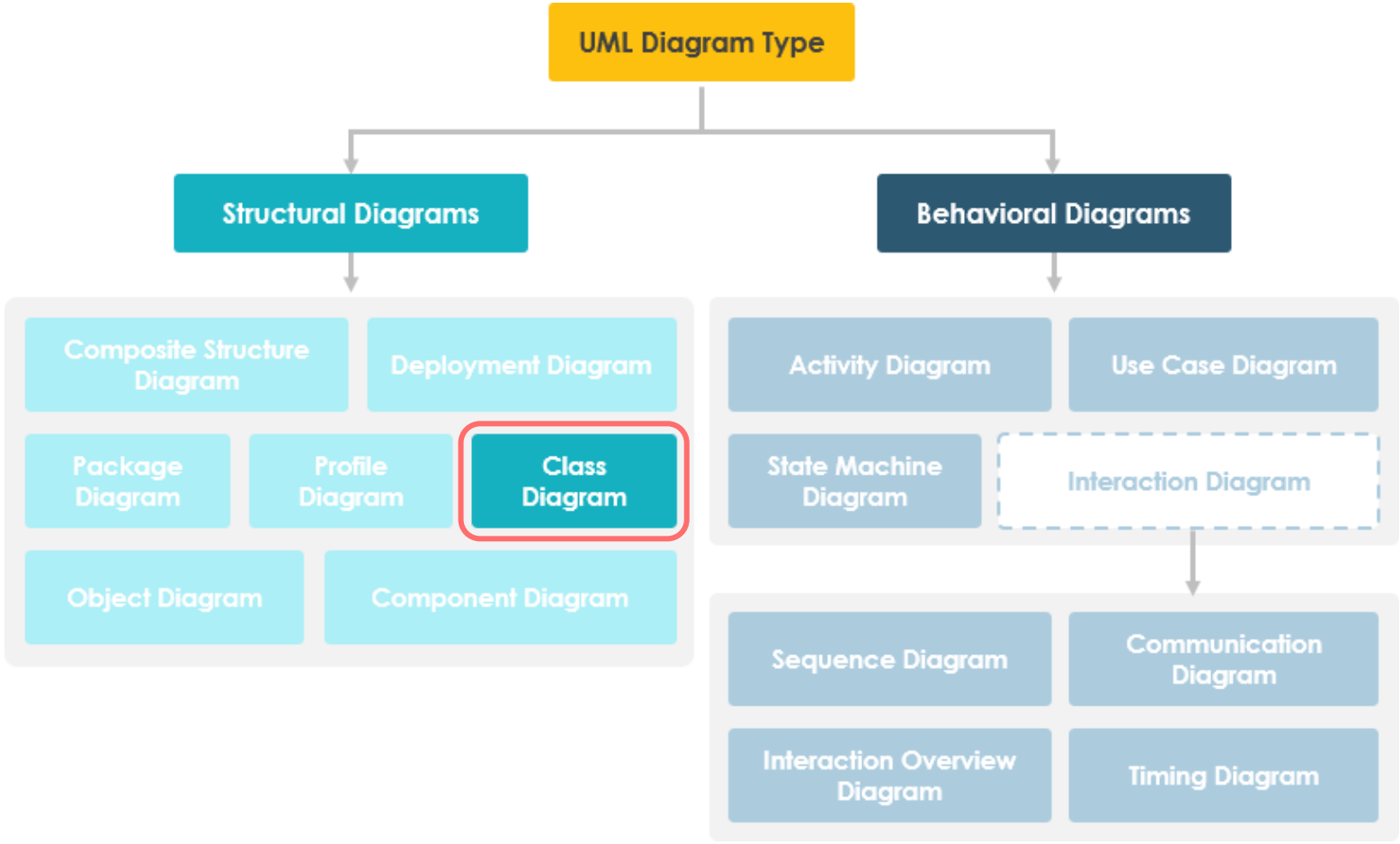
Inxhinieria **Softuerike**

Faza: Dizajni i Modelit të Klasës (UML Diagrami i Klasës)

Ramiz HOXHA
ramiz.hoxha@ubt-uni.net
2020/2021

UML definon dy lloj të Diagrameve

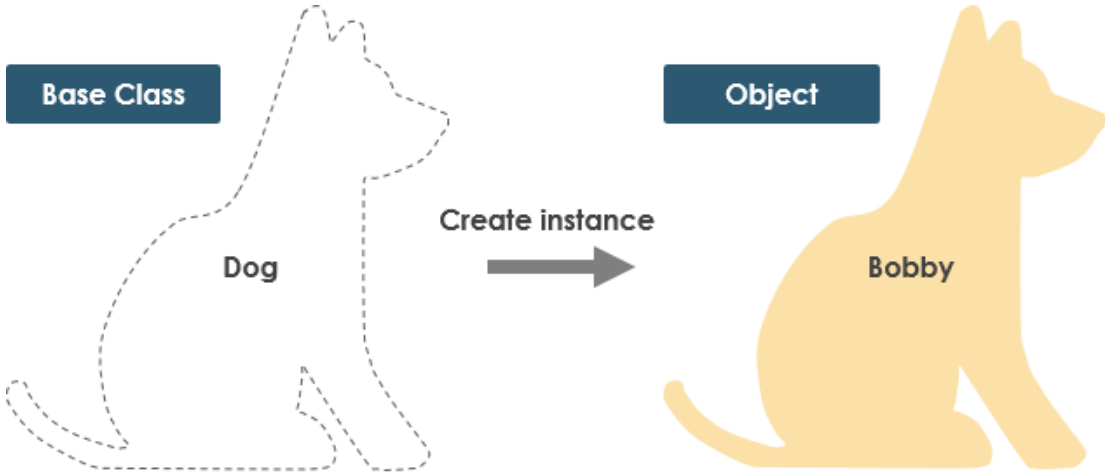


Çfarë është një Klasë

- Një Klasë është një plan (template) për objektin. E gjithë qëllimi i Dizajnit të Orientuar në Objekt (OOD) nuk ka të bëjë me objektet, ka të bëjë me klasat, sepse ne përdorim klasa për të krijuar objektet.

- Në fakt, klasat përshkruajnë llojin e objekteve, ndërsa objektet janë raste të instancave të klasave.
  - *p.sh*: Një **qen** ka **atributet** – ngjyra, emri, raca, si dhe **sjelljet** – hece(), shkunet(), leh(), han().

Një ***object*** është një instance e **Klasës**



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

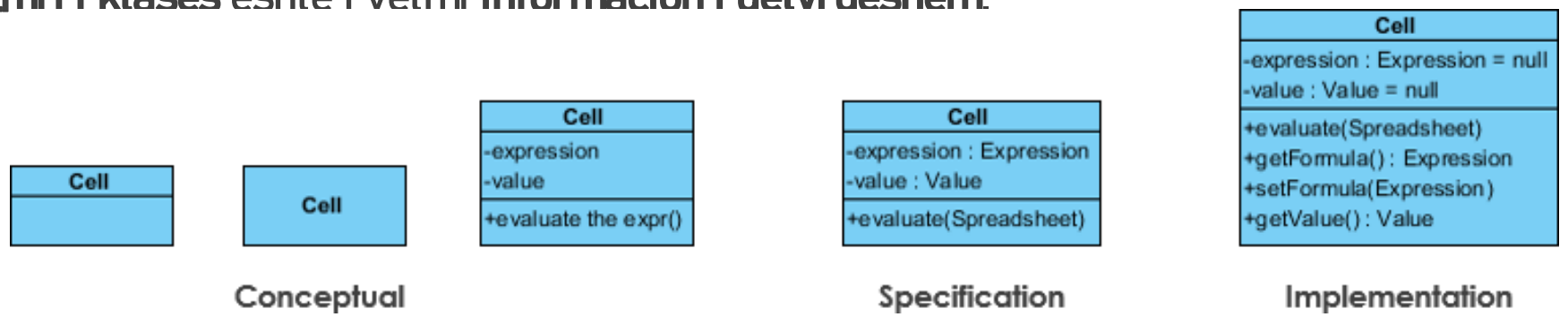
Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight: 24 pounds	

Çfarë është një Klasë…

- Një Klasë është një përshkrim i një grupi objektesh të gjitha me role të ngjashme në sistem, i cili përbëhet nga:
- Vetit strukturore (atributeve) definojn se çka mund të 'i dim' për objektet e klasës
  - **Reprezentojn gjendjen** e një objekti të **klasës**
  - Janë **përshkrime** të **veçorive strukturore** ose **statike** të një **klase**
- Sjelljes (operacione) definojn se çka "mund të bëjën" objektet e klasës
  - Definojnë **mënyre** në të cilen objektet mund të **ndërveprojn**
  - **Operacione** janë **përshkrime** të **sjelljeve** ose karakteristika **dinamike** të **klasës**

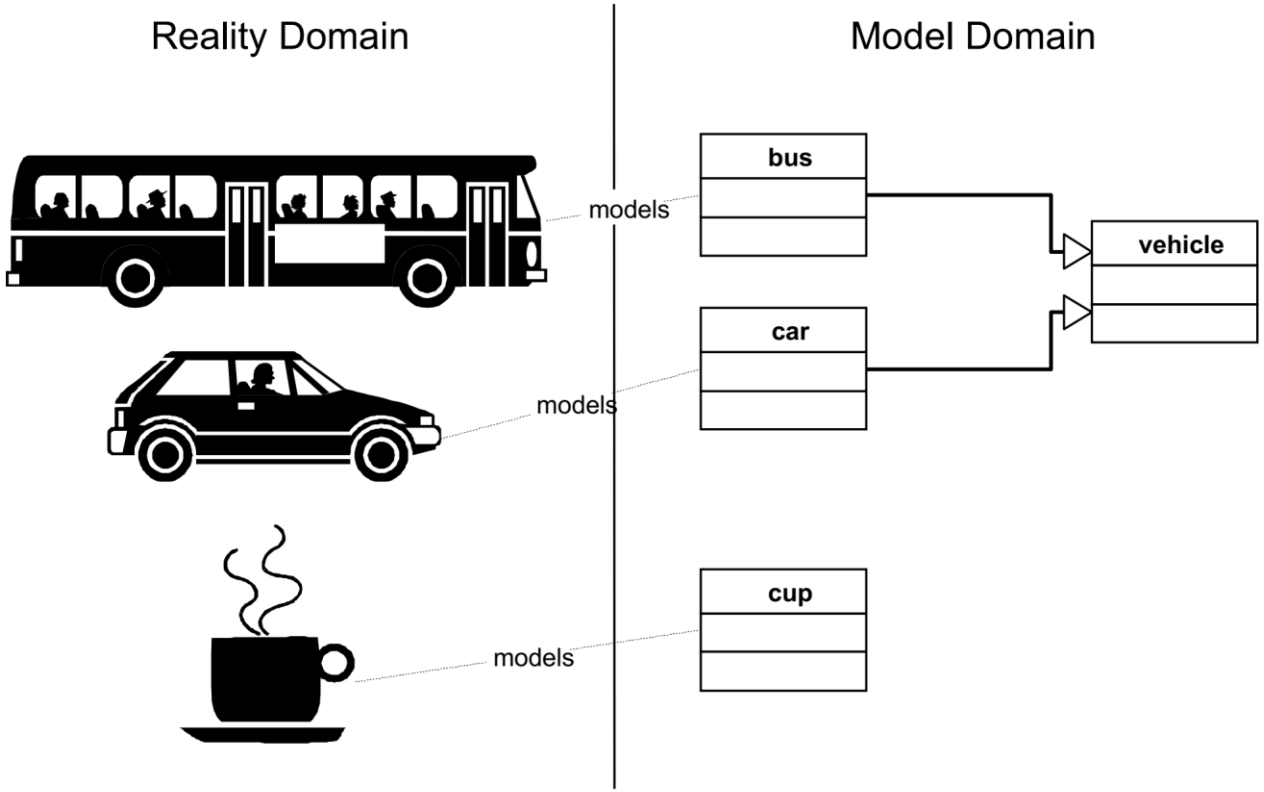
Përspektivat e Diagramit të Klasës në UML

- Një diagram mund të interpretohet nga këndvështrime të ndryshme:
  - Konceptual: **paraqet** konceptet **Objektet** e **domenit** (Klasa e Domaint)
  - Specifikimi: **fokusi** është në **interfaces** e të dhënave abstrakte **Objektet e Interfacave** (ADTs) në softuer.
  - Implementimi: **përshkruan** se si **klasa** do të implementoj **interface** e tyre, **Objektet e Entiteteve**
- Perspektiva ndikon në sasinë e detajeve që duhen furnizuar dhe llojet e relacioneve që ia vlen të paraqiten.
  - **Emri i klasës** është i vetmi **informacion i detyrueshëm**.



qasja e Orientuar në Objekte (OO Aproach)

- **Objektet janë abstrakte të botës reale ose entitetet të sistemit**



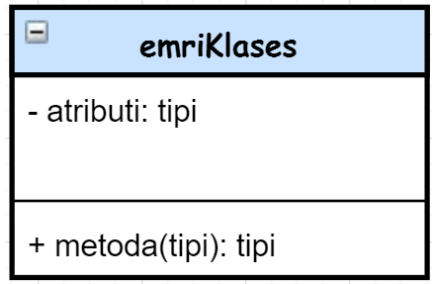
Diagrami i Klasës në UML

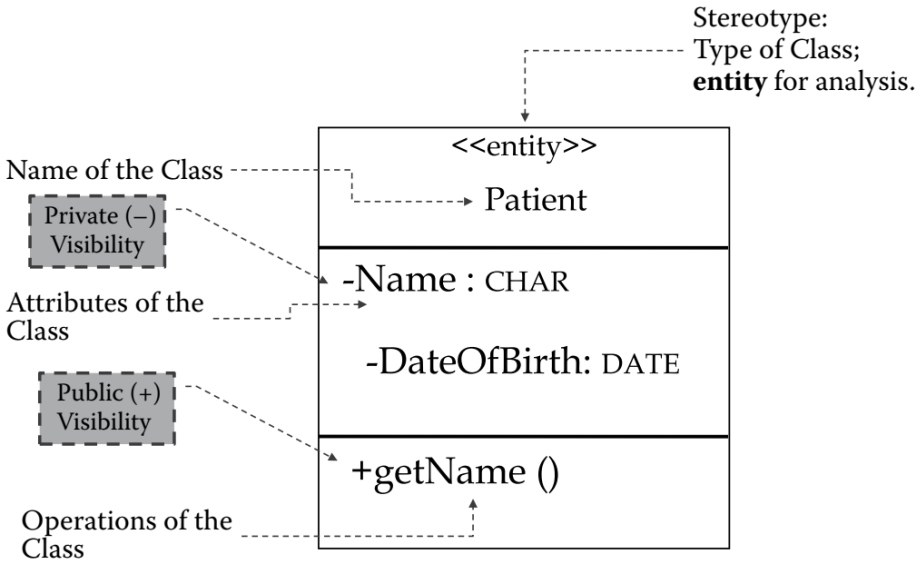
- Në inxhinieri softuerike, një diagrami i klasës në UML (Gjuhë e unifikuar për modelim) është diagrami që paraqet aspektin statik që përshkruan strukturën e sistemit duke paraqitur klasat e sistemit, atributet e tij, operacionet (ose metodat), dhe lidhjet/relacionet ndërmjet objekteve.
  - **Diagrami i klasëve**, ilustroni **modelet e të dhënave** për **sistemet e informacionit**, pa marrë parasysh sa e **thjeshtë** ose **komplekse** janë.
  - **Klasat** përfaqsojnë **entitetet e botës reale** ose **koncepte të sistemit**.

Klasa e objekteve - një grup objektesh që ndajnë atribute dhe sjellje të përbashkëta, nganjëherë i referohemi si një klasë.

Reprezentimi i një Klasë (simboli).

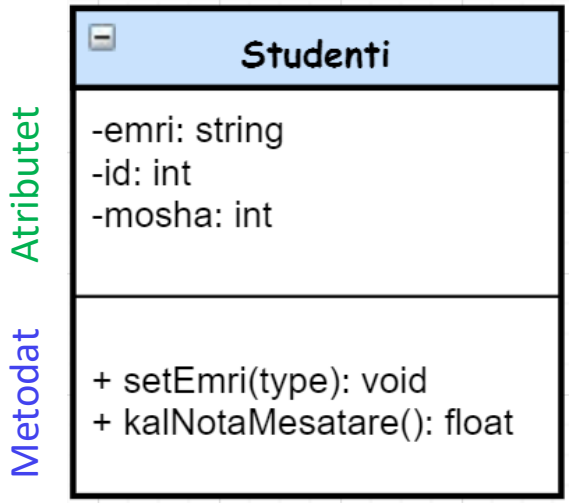
- Një klasë përshkruan **një grup objektesh** që kanë:
  - **karakteristika/veti** të **ngjashme** (atributet),
  - **sjellje** të **ngjajshme** (operacione),
  - **relacione** të **përbashkëta** me objekte të tjera, dhe
  - **kuptim** të përbashkët ("semantikë").
    - Atributet: një pjesë e rëndësishme e të *dhënave* që përmbajnë *vlera* që përshkruajnë secilen *instance (karakteristikat)* të klasës.
    - E quajtur edhe *fusha*, *variabla*, veti.
    - Metodat: gjithashtu i quajtur operacion ose funksione.
    - Ju lejojnë të specifikoni ndonjë funksion të sjelljes së klasës.





Diagrami i Klasës në UML: Rasti i kampusit të UBT's

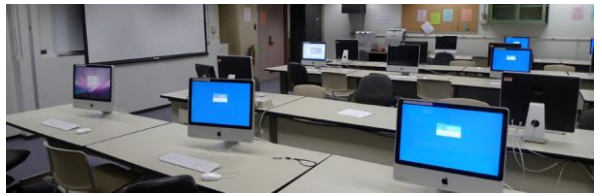
- Rasti i studimit të kampusi Universitar (Kolegji UBT).
  - Ne kemi nevoj të i **paraqesimi (reprezentojm) gjërat** e ndryshme që **janë në sistem**, këte mund te e **bejme** duke **përdorur klasët**.
  - Pra, çfarë ka në kampusin e një universiteti, ka shumë **student, profesor, departamente, puntor administrative, objekte/ndërtesat, salla labaratorike**, etj.



- **Atributet**: një pjesë e rëndësishme e të ***dhënave*** që përmbajnë ***vlera*** që përshkruajnë secilen ***instancë*** të klasës.
- E quajtur edhe ***fusha***, ***variabla***, veti.

- **Metodat**: gjithashtu i quajtur operacion ose funksione.
- Ju lejojnë të specifikoni ndonjë funksion të sjelljes së klasës.

studenti



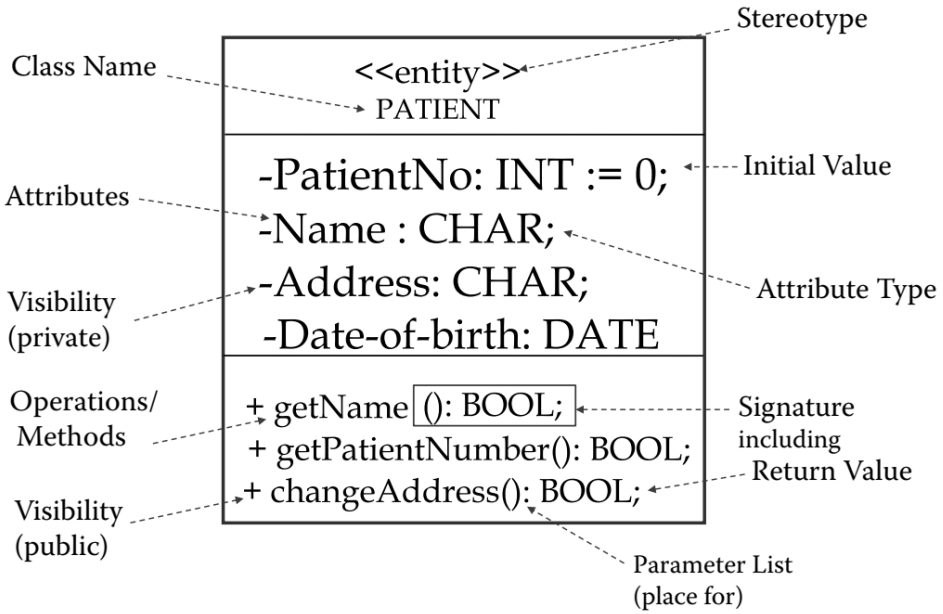
puntori administrativ

Profesori

Ndërtesat/objektet

Shembulli i Kodit i cili pasqyron klasën e Pacientit të treguar me poshte.

ky është një **pseudo-kod** dhe jo në një gjuhë specifike.



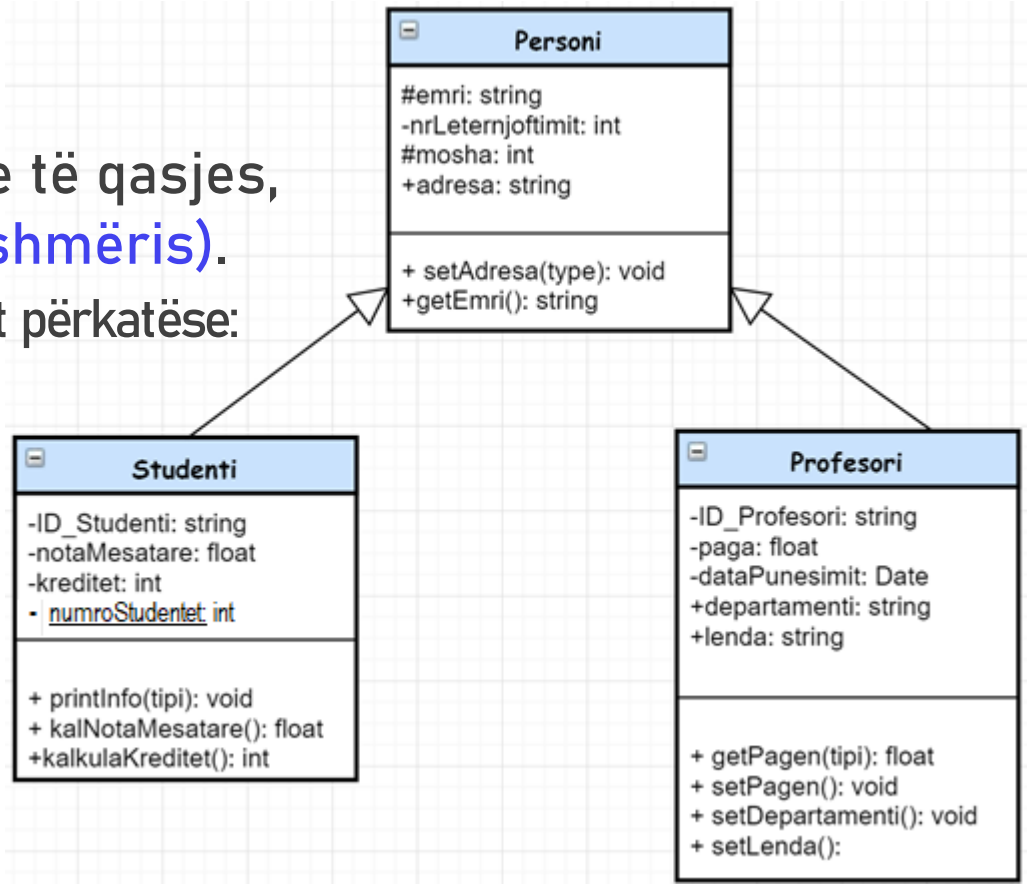

```
class PatientDB
{
 public int saveDetails(int p, String n, String a, Date d)
 {
 return -1;
 }
}
class Date
{
 }
```

```
class Patient
{
 // private attributes
 private int PatientNo;
 private String Name;
 private String Address;
 private Date DateOfBirth;

 // public methods
 public boolean getName(String PatientName)
 {
 // code here
 return false;
 }
 public boolean getPatientNumber()
 {
 //code here
 return false;
 }
 public boolean changeAddress(String NewAddress)
 {
 this.Address = NewAddress;
 return true;
 }

 public int saveChanges()
 {
 int ReturnValue;
 PatientDB db = new PatientDB();
 // pass the current values of the attributes of the "this" Patient
 // the next line calls a database module to save the details of
"this" Patient to the DB.
 // Return Values will identify success or reasons for failure.
 ReturnValue = db.saveDetails(PatientNo, Name, Address,
DateOfBirth);
 // generally, if ReturnValue is not 0 then there was an error.
 // It is upto the calling class to deal with this in an appropriate
manner.
 // e.g. notify the user that there was an error, log to error log,
etc.
 return ReturnValue;
 }
}
```

Diagramet e Klases në UML: Vizibilitetit në një klasë

- Specifikuesit e qasjes (dukshmëria) së anëtarëve/variableve.
- Të gjitha klasat kanë nivele të ndryshme të qasjes, varësisht nga modifikuesi i qasjes (dukshmëris).
  - ✓Kemi nivelet e mëposhtme të **qasjes** me simbolet përkatëse:

Qasja në Klasë (OOP)	UML simbole
Qasja public	+
Qasja private	-
Qasja protected	#
Qasja package	~
Qasja static	nënvizuar

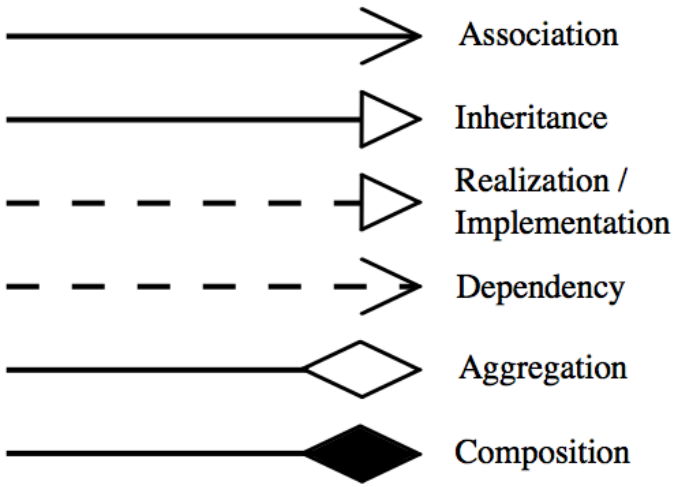


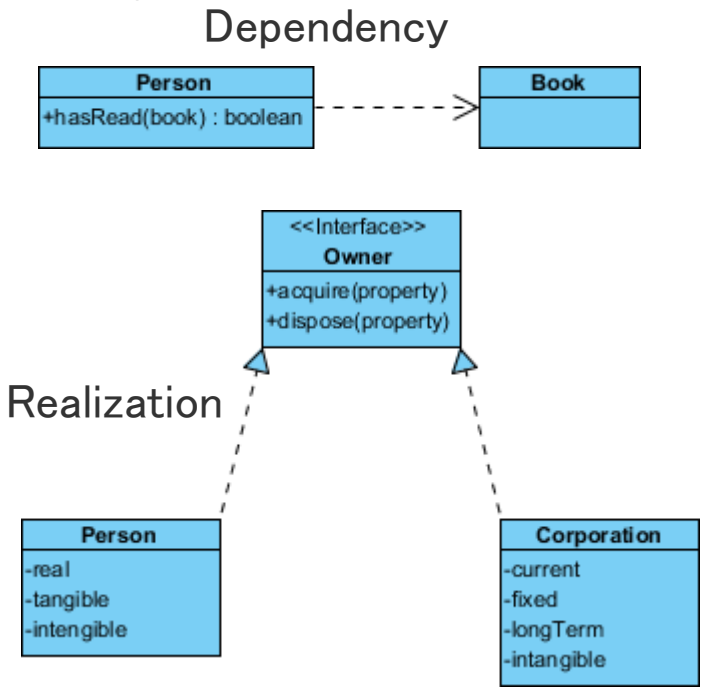
Diagramet e Klases në UML: Vizibilitetit në një klasë...

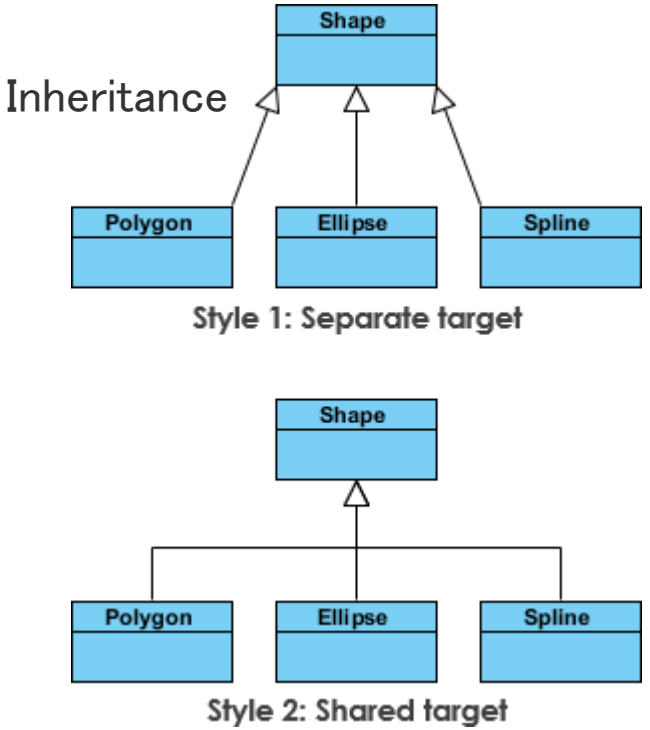
Qasja në Klasë (OOP)	UML simbolet	
public	+	Anëtarët (***atributet***) e të dhënave publike janë të **qasshëm** *brenda* klasës, *jashtë* klasës, nga *çdokush*
private	-	Anëtarët (***atributet***) e të dhënave private janë të **qasshëm** *vetëm brenda* klasës në të cilën janë *dëfinuar*
protected	#	Variablat e mbrojtura (protected) janë atributet të qasshëm *brenda* klasës në të cilën janë *definuar* dhe nga të gjitha klasat e *fëmijëve* të klasës
packege	~	Anëtarët e të dhënave package mund të janë të qasshëm *brenda* klasës në të cilën janë *definuar* dhe gjithashtu *brenda të gjitha* klasave në atë *paketë*
static	nënvizuar	Anëtarët e të dhënave statike janë atributet që *shfrytzohet (ndahen)* nga të *gjithë objektet e klasës*. Ekziston vetëm një kopje e anëtarëve të të dhënave statike

Diagramet e Klasës në UML: Relacionet

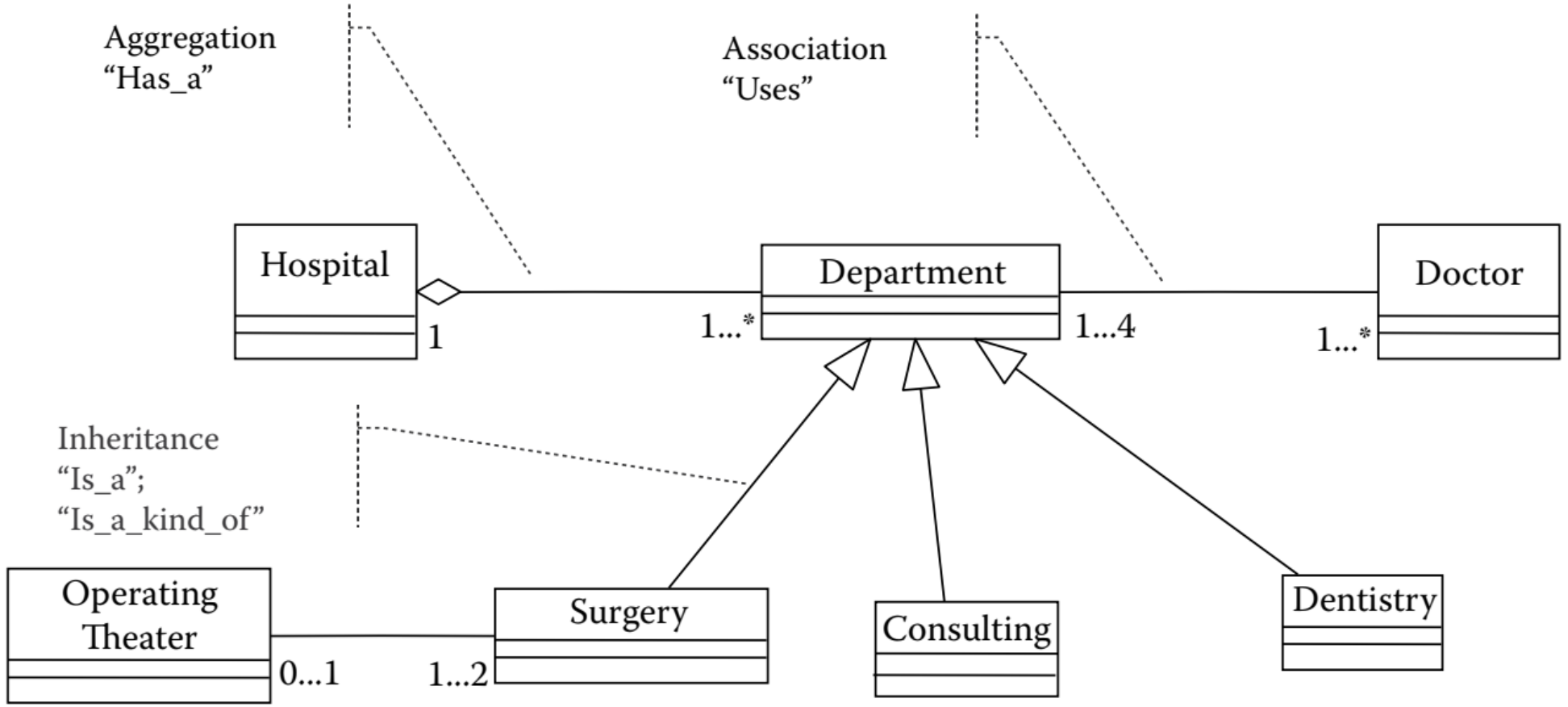
- Një relacion është një term i përgjithshëm që mbulon llojet e veçanta të lidhjeve logjike të gjetura në diagramet e klasës dhe objekteve.
- UML përcakton relacionet e mëposhtme:







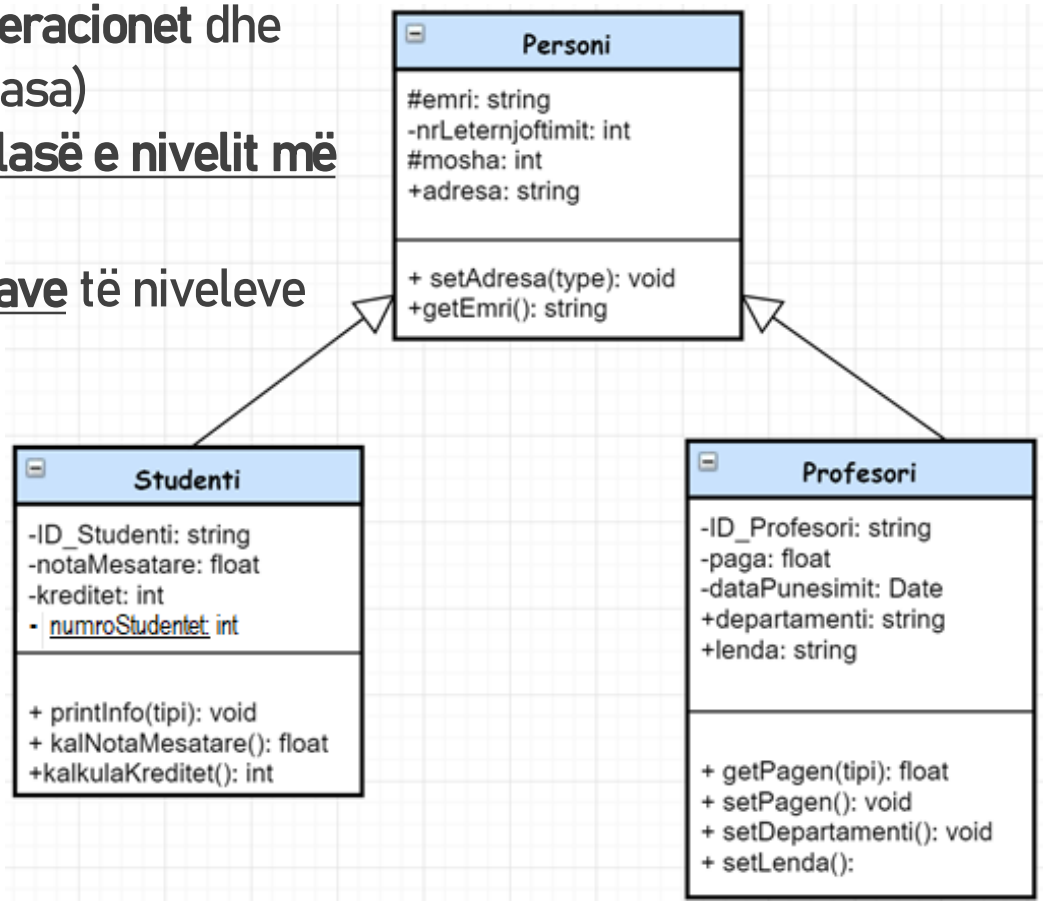
Shembull: Relacionet në një diagram të klasës.



Diagramet e Klasës në UML:

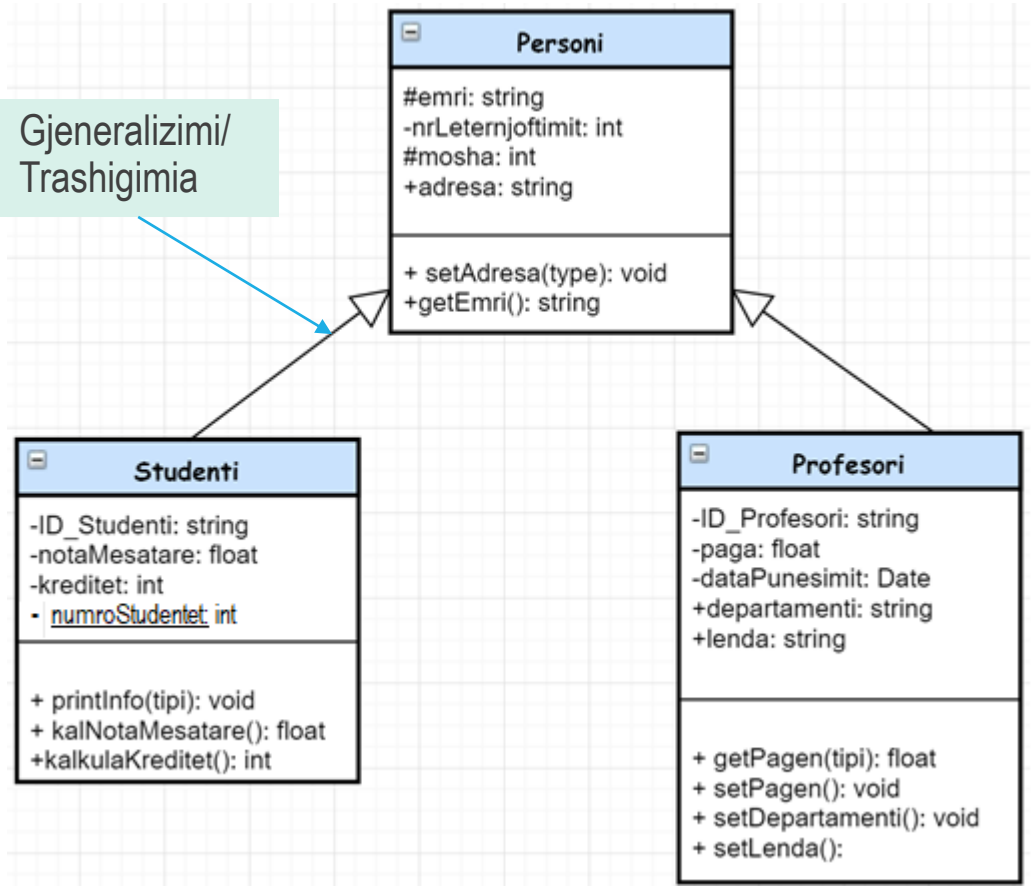
Relacioni i gjeneralizimi/trashigimia

- **Trashëgimia/Inheritance** nënkupton që **atributet, operacionet** dhe relacionet e një **klase të nivelit më të lartë** (superklasa) trashëgohen nga-të bëra në dispozicion - nga **një klasë e nivelit më të ulët** (nënklasa).
- Kështu, klasat e nivelit më të ulët **janë "lloj" të klasave** të niveleve më të larta .

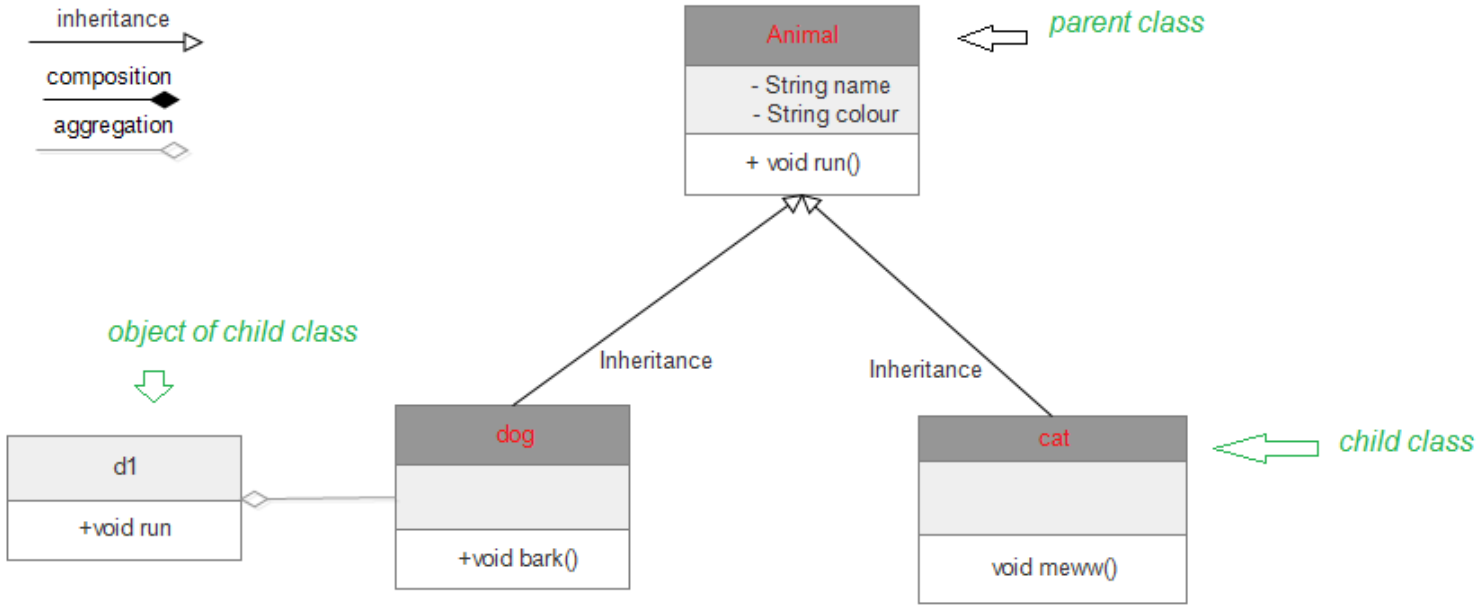


- Ky **relacion** është *specifike* për një qasje të **orientuar drejt objektit** për zhvillimin e **sistemeve softuerike.**

Relacioni i **gjeneralizimi/trashigimia**



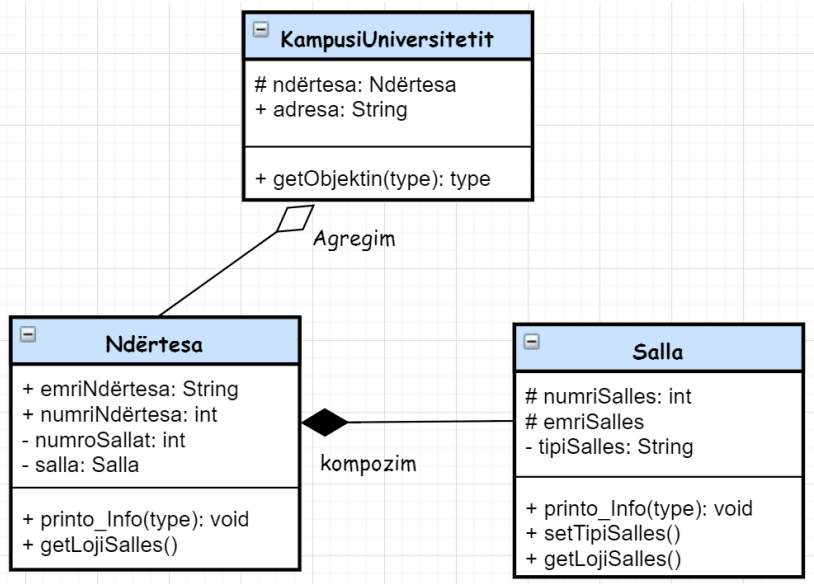

```java
public class Personi {
    protected String emri;
    private int nrLeternjoftimit;
    protected int mosha;
    public String adresa;


    public void setAdresa(String adresa) {
        this.adresa = adresa;
    }
    public String getEmri() {
        return emri;
    }
}
public class Studenti extends Personi {
    private String ID_Studenti;
    private float notaMesatare;
    private int kreditet;
    private static int numroStudentet;

    public void printInfo() {
        System.out.println();
    }
    public float getkalNotaMesatare() {
        return notaMesatare;
    }
    public void setInfo(String IDStudenti, String emri, float nm ) {
        this.ID_Studenti = IDStudenti;
         this.emri = emri;
         this. notaMesatare= nm;
    }
}
```




```java
class GFG {
    public static void main(String[] args)
    {
        dog d1 = new dog();
        d1.bark();
        d1.run();
        cat c1 = new cat();
        c1.meww();
    }
}

class Animal {
    public void run()
    {
        String name;
        String colour;

        System.out.println("animal is running");
    }
}

class dog extends Animal {
    public void bark()
    {
        System.out.println("wooh!wooh! dog is barking");
    }
    public void run()
    {
        System.out.println("dog is running");
    }
}

class cat extends Animal {
    public void meww()
    {
        System.out.println("meww! meww!");
    }
}
```

Diagramet e Klasës në UML:
Agregimi & Kompozimi

- Relacioni i agregimit në diagram të klasës.
  - nëse relacionet midis dy klasave asocimi është i ngushtë, atëherë ajo përfaqësohet nga agregimi, është kështu një formë e veçantë e asociimit.
  - Agregimi reprezenton një klasë që përmban një klasë tjetër.
    - Fig, tregon relacionin "has a" ku një Kampusi ka Ndërtesa.
  - Agregim gjithashtu reprezenton **kompozim** d.m.th, një klasë e përbërë nga një klasë ose shumë klasë tjetëra.
  - P.sh, një **ndertesë** përbëhet apo kompozohet nga sallat (klasat që përbëjnë ndërtesen)

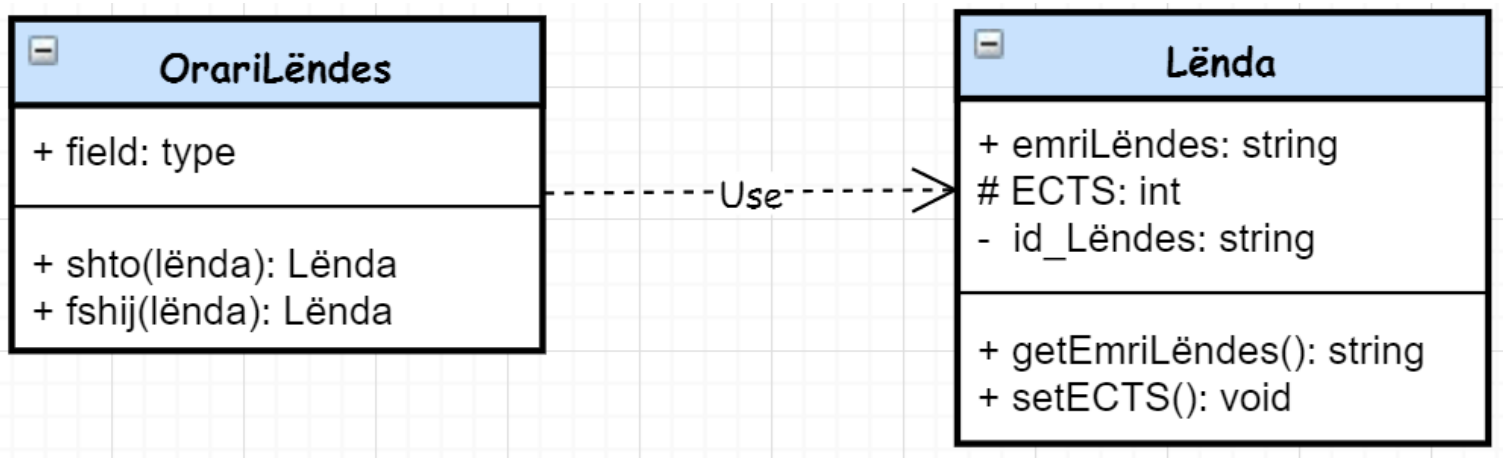



```
public class Salla {
}
public class Ndertesa {
     private Salla[] salla;
}
public class KampusiUniversitetit{
    private Ndertesa ndertesa;
}
```

Diagramet e Klasës në UML:

Relacioni i varësisë (Dependency)

- Një relacioni i varësiës tregon një *relacion semantike* midis dy ose më shumë elementeve.
  - Vartësia e **OrariLëndes** në **Lënden** ekziston sepse **Lënda përdoret** edhe për *operacionin* **futjes/shtimit** dhe **fshirjen/heqjes** të orarit të lëndes (**OrariLëndes**).

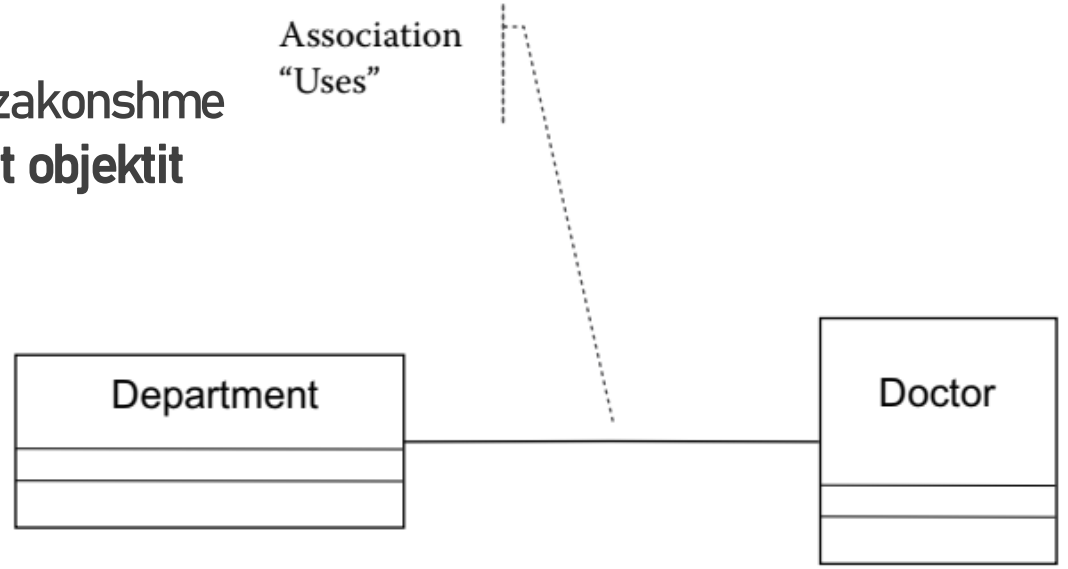


Diagramet e Klasës në UML: Asocimi

- Relacioni i asocimit në UML Diagram Klasës.
  - Nëse dy klasa në një model duhet të **komunikojnë** me **njëri-tjetrin**, duhet të ketë **lidhje midis** tyre. Një **asocim** tregon atë **lidhje**.
  - **Asocimi** ndërmjet dy klasave **tregon si objektet** në një anë të një **asociacioni** **"njohin" objekte** në **anën tjetër** dhe mund të dërgojnë mesazhe tek ata.
  - Nëse një asocim është **drejtuar**, mesazhet mund të kalojnë **vetëm në atë drejtim**
  - Nëse asocimi nuk **ka drejtim(direkcion),** atëherë lidhja është **definuar** si një **lidhje dydrejtimesh** dhe mesazhet mund të kalojnë në të dy drejtimet
  - Sipas paracaktimit (**default**), të gjitha relacionet duhet të drejtohen, përveç **nëse kërkesat** kërkojnë lidhje **dydrejtimesh**.

Diagramet e Klasës në UML: Asocimi...

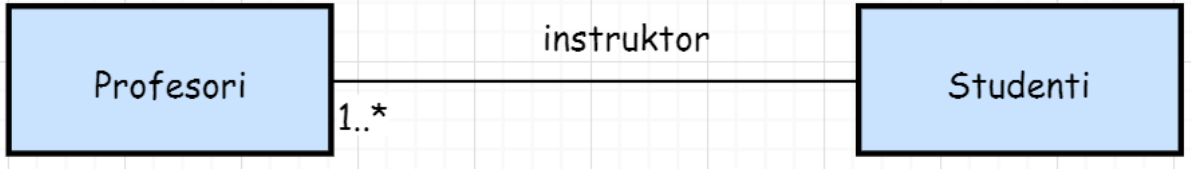
- Fig tregon dy klasa, Departamenti dhe Doktori, asocohen me njëra-tjetrën.
  - Të dy klasat e departamentit dhe doktorit “use" njëra-tjetrën.
  - Një **asocim** është **relacion më bazik** dhe më i zakonshme midis dy klasave në modelet e **orientuara drejt objektit**



Doctor *uses* Department; Department also uses Doctor; The Relationship is *loose*.

Diagramet e Klasës në UML: Asociacionet & pjesamarrja

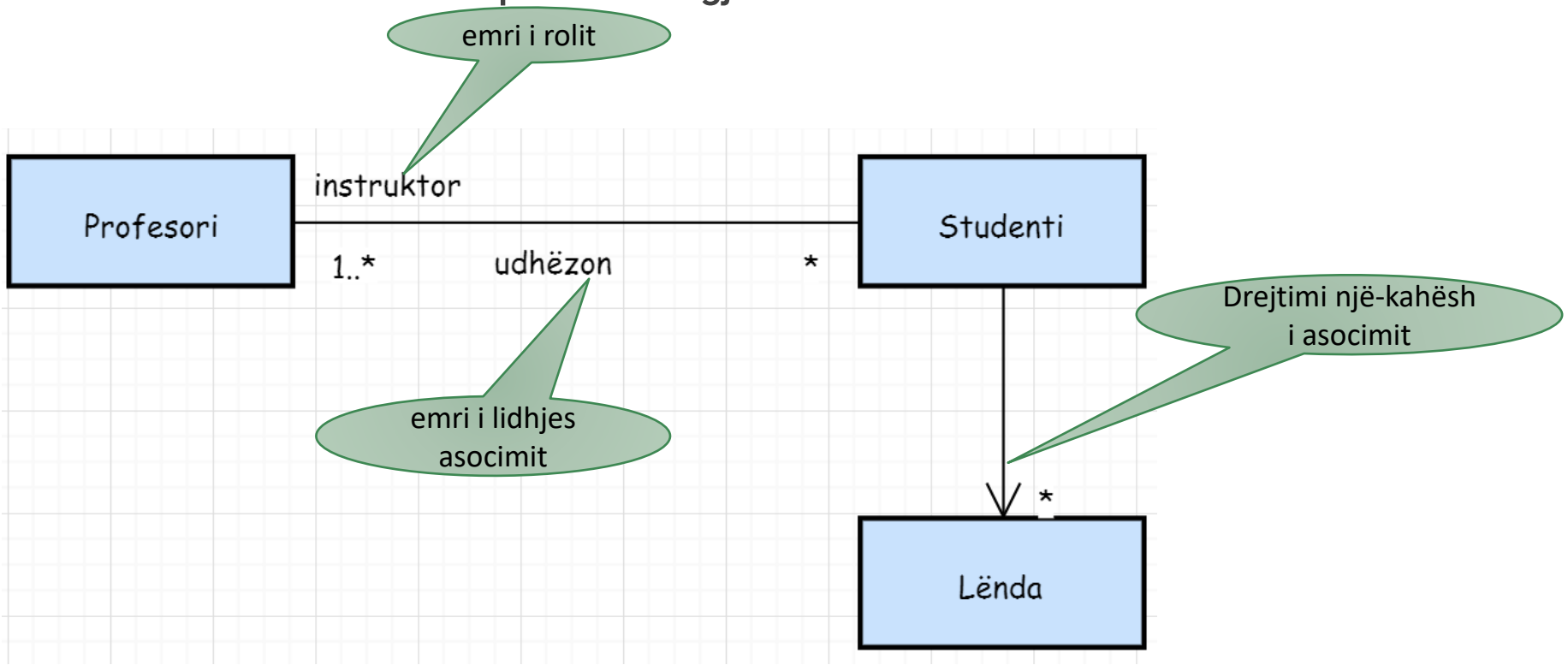
- Ne mund të tregojmë pjesmarrjen e një asocimi duke shtuar pjesmarjen e shumëfishta në vijën që tregon lidhjen.
  - *Numrat prane* **klasave** tregojne se sa *është pjesmarrja* ne këtë lidhje.
  - Shembulli tregon kur **një student** ka **një** ose më **shumë instruktorë**:



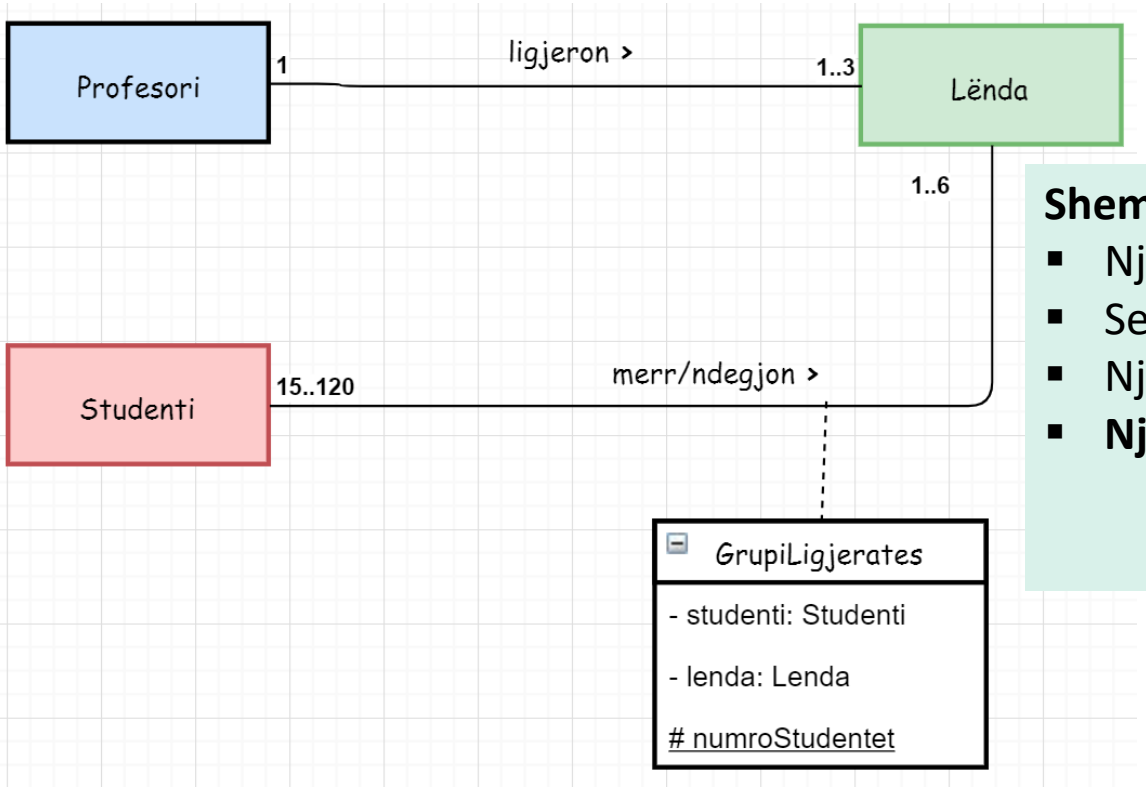
Vlera e tregusit në pjesmarrje	Simbolet e vlerave te pjesmarrjes
0	zero/asnje instance
1	një instance të vetëm
*****	Vlere e pakufizuar int jo negative e instancës
0..1	zero 0 ose një instance të vetme
0..*	zero 0 ose numer të pakufizuar (shume) të instancës
1..*	një ose më shumë instanca
3..6	Rang i specifikuar

Diagramet e Klasës në UML: Asociacionet & pjesamarrja

- Shembulli 1: Tregon kur min ***një profesor*** ose më shumë ***udhëzon/udhëzojnë*** shumë **student**, po ashtu **studentët** kan shumë **lëndë** për të i ndegjuar.

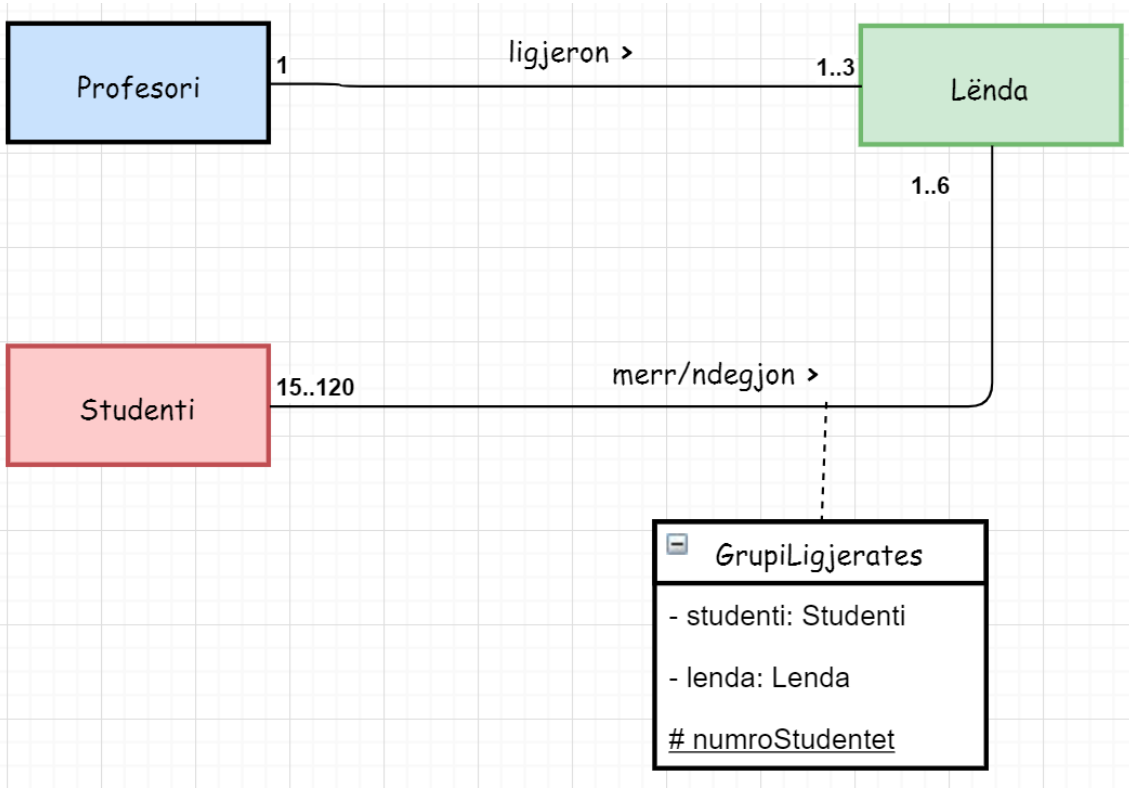


Diagramet e Klasës në UML:
Asociacionet & pjesamarrja



Shembulli 2:

- Një **profesor** *ligjeron* 1 deri në 3 **lëndë**
- Secila **lënd** *ligjerohet* nga **vetëm një profesor**.
- Një **student** mund të *merr/ndegjojë* 1-6 **lëndë**.
- **Një lëndë** mund të ketë 15-120 **student**.

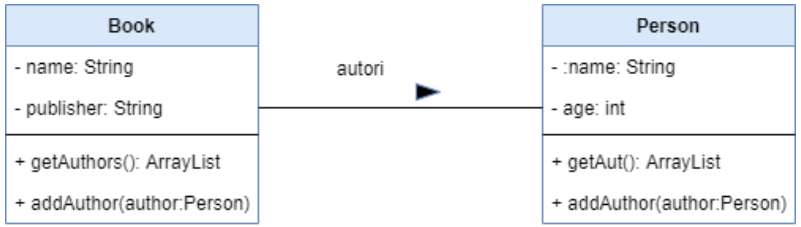



```java
public class Profesori {
private String ID_Profesori;
Private String emri;
private float paga;
}
public class Studenti {
private String ID_Studenti;
Private String emri;
private  int kredit;
}

public class Lenda {
    private String emriLendes;
    private Profesori ligjerusi;
}

public class GrupiLigjerates {
    private Lenda[] lende;
    private Studenti[] student;
    private Lenda[] profesori;

    ArrayList<Lenda> lende = new ArrayList<Lenda>();
    ArrayList<Studenti> Student = new ArrayList<Studenti>();

    public GrupiLigjerates(){
        lende = new Lenda[6];
        student = new Studenti[120];

        .....
    }
}
```

Asocim bazik vs Dependecy

Asocim bazik

- Shigjeta tregon drejtimin e lidhjes. Lidhja tregon se Libëri e njeh autorin e tij, por një Person nuk di për librat që ata janë autor.



Krijon asocim/lidhje bazike

```
public class Book {
 private String name;
 private String publisher;
 private ArrayList<Person> authors;

 // constructor


 public ArrayList<Person> getAuthors() {

            return this.authors;
            }

 public void addAuthor(Person author) {

            this.authors.add(author);
            }
 }
```

```
public class Person {
private String name;
private int age;

public Person(String initialName) {
            this.name = initialName;
            this.age = 0;
            }

public void printPerson() {
            System.out.println(this.name + ", age " + this.age + " years");
            }

public String getName() {
            return this.name;
            }
}
```

Asocim bazik vs Dependency

- Dependecy/Varesi
  - Lidhja e tregon se ekzistenca e klases Libri është e varur nga klasa Person

Kufizohet sipas dependency

```
public class Book {
 private String name;
 private String publisher;
 private ArrayList<Person> authors;

 // constructor
 Public Book (ArrayList<Person> Authors, String name, String publisher ){
            this.authors= Authors;
            this.name= name;
            this.publisher= publisher;
            }

 public ArrayList<Person> getAuthors() {

            return this.authors;
            }

 public void addAuthor(Person author) {

            this.authors.add(author);
            }
}
```

```
public class Person {
private String name;
private int age;

public Person(String initialName) {
            this.name = initialName;
            this.age = 0;
            }

public void printPerson() {
            System.out.println(this.name + ", age " + this.age + " years");
            }

public String getName() {
            return this.name;
            }
}
```

Identifikoni klasat / entitetet e biznesit

- Identifikimi i klaseve:
  - Analizo User Stories (tregimet e perdoruesit)
  - Identifiko klasat
    - Emrat → Klasa
    - Mbiemrat → Attribute
    - Foljet → Operacione (funksione, metoda)

Identifikoni klasat / entitetet e biznesit….

- **Me nënviza identifikohen `emrat`, `mbiemrat` dhe `foljet` nga User Stories.**

Card:

Une si `anëtarë i familjes` duhet të kem mundësi të `regjistrimin` e `produkteve` ne sistem.

Conversation:

1. Kush ka mundësi ti regjistroj produktet?
   Secili `anëtar i familjes` i cili ka llogari ne sistem mund të `shtoj produkte`.
   Çfarë informata i nevojiten për `produktin` ?
   Nevojiten të jepen `emri i produktit`, `data e skadimit`, `kategoria` dhe `çmimi`.
2. Ku realizohet ky proces?
   Procesi realizohet ne mobile aplikacion dhe pastaj `shtohet` ne sistem.

Confirmation:

Given: `Anëtari i familjes` është i regjistruar në sistem (ka llogari)
When: `Anëtari i familjes` `kyçet` në sistem
And: Zgjedh opsionin ‘`Regjistro produktin`’
Given: Hapet dritarja ku plotësohen `të dhënat` për `produktin`
Then: Kur të dhënat e produktit `shtohen` si `emri`, `data e skadimit`, `kategoria` dhe `çmimi`.
When: Anëtari i familjes klikon butonin ‘Ruaj’
Then: `Produkti` `regjistrohet` në sistem

And: `Anëtarit` të familjes i bëhet konfirmimi që `produkti` është ‘Regjistruar me sukses’

Identifikoni klasat / entitetet e biznesit….

Card:

Une si anëtar i familjes duhet të kem mundësi të shtimit të produkteve ne frigorifer.

Conversation:

1. Cilat produkte mund t'i shtojmë në frigorifer?

   Poroduketet të cilat janë paraprakisht të regjistruara.

1. Kush mund të bëjë shtimin e produkteve në frigorifer?

   Të gjithë anëtarët e familjes të cilët kanë llogari në sistem

1. Si evidentohet shtimi ne frigorifer?

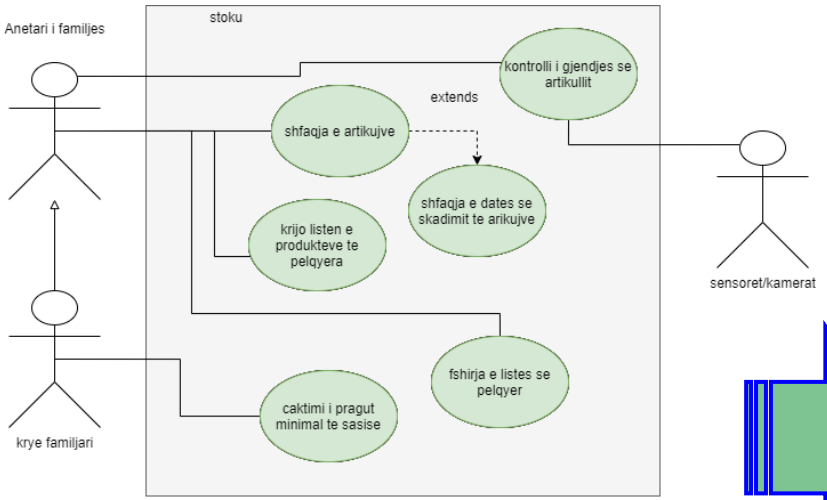
   Sensoret ne dere të frigoriferit skanojnë kodin e produktit, e konverton ne një identifikues, pastaj e shton produktin ne frigorifer dhe e shton sasinë përkatëse.

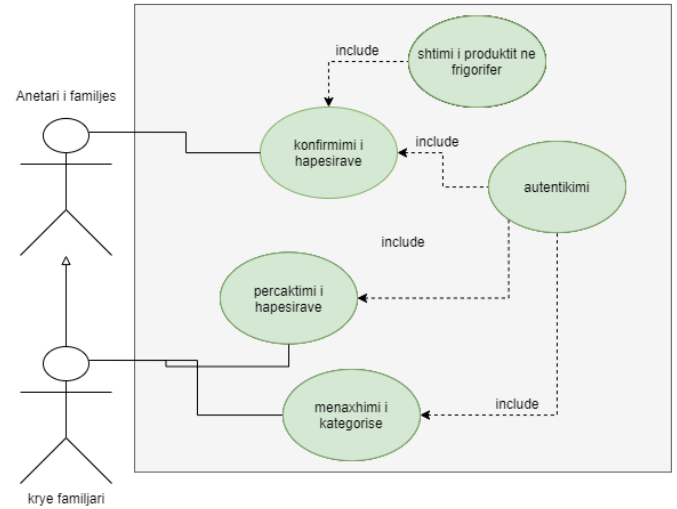
Confirmation:

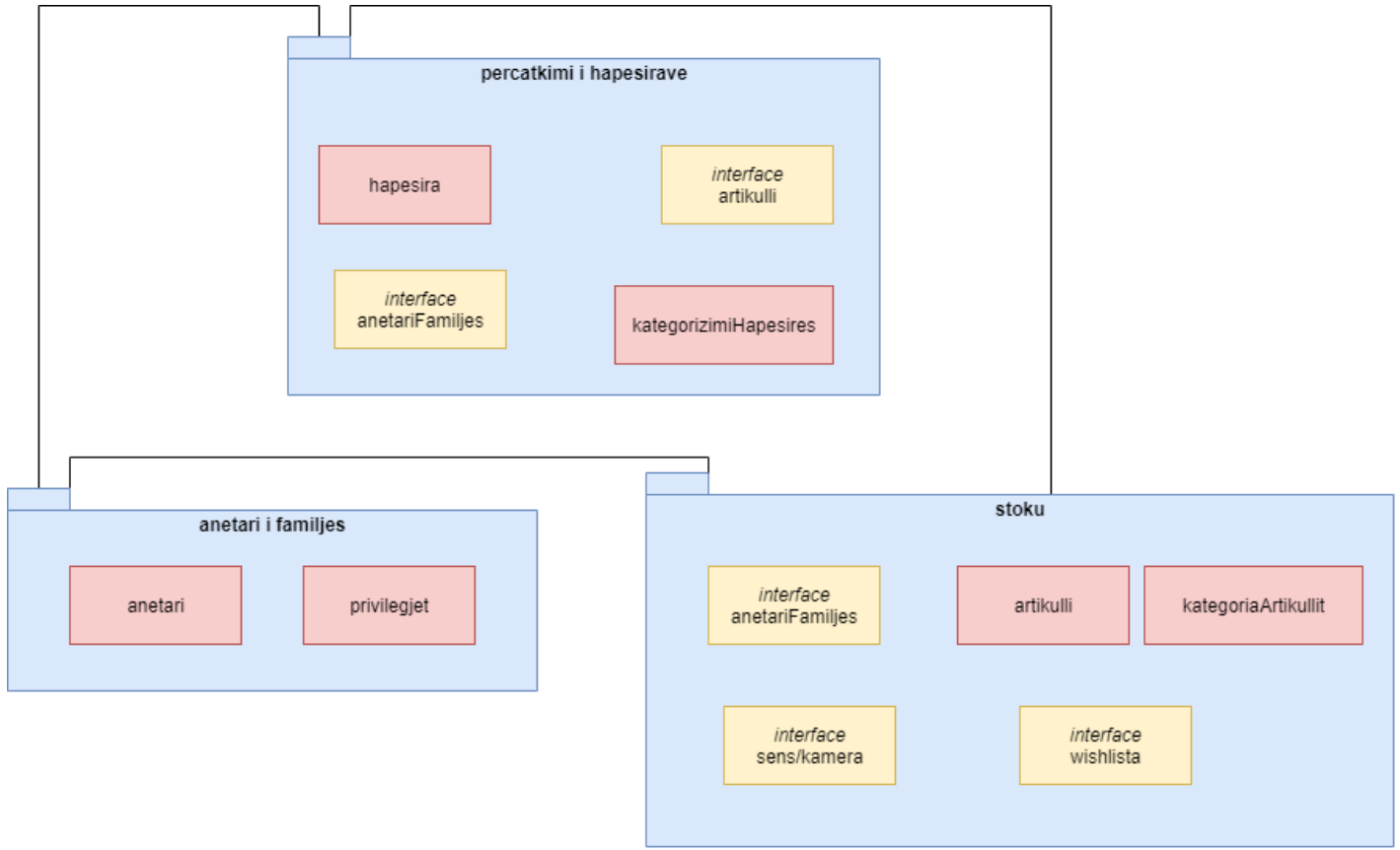
Given: Anëtari i familjes është i regjistruar në sistem (ka llogari)
When: Anëtari i familjes kyçet në sistem
And: Zgjedh opsionin 'Shto produktin'
Given: Hapet dritarja ku shfaqen të gjitha produktet të cilat janë të regjistruara paraprakisht.
Then: Anëtari i familjes zgjedh produktin përkatës
Given: Sensorët bëjnë leximin (skanimin) e barkodit
Then: Shfaqen të dhënat për produktin
Then: Anëtari i familjes konfirmon produktin e shfaqur

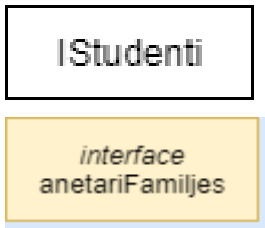
Then: Produkti shtohet në frigorifer

Dizajni i disa Moduleve për sistemin **SmartFridge**





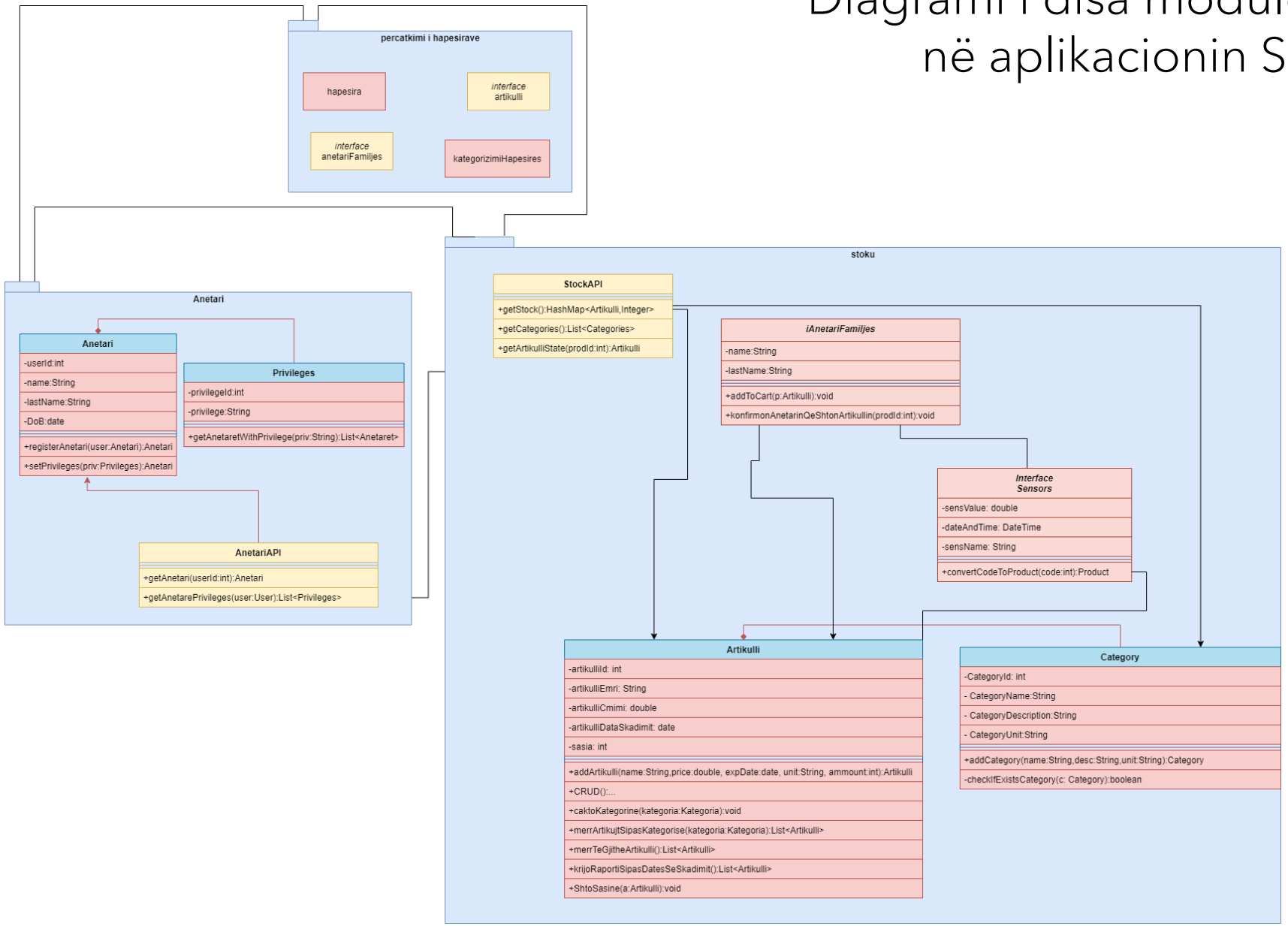




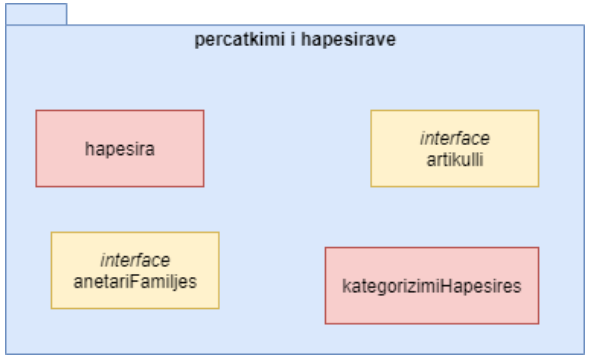
- p.sh *Interface antariFamiljes/IStudenti* d.m.th me simbolin referohemi interface se Entiteteve në vend të moduleve (entiteteve) konkrete
- nënkupton klasat/objektet /entitetet/ që implementojnë interface
- Nuk është një interface mbrenda Modulit përkates

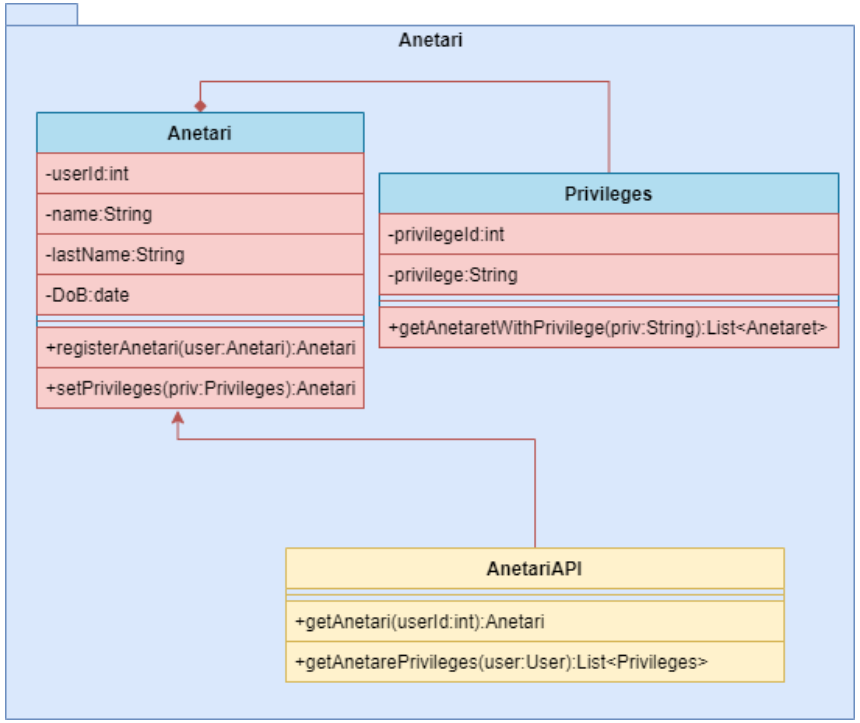
Për t'i qasur **metodatve e interfaces**, interfaca duhet të "*implementohet*" nga një klasë tjetër me fjalën kyçe **`implements`** dhe metodat duhet të implementohen në klasën e cila trashëgon vetitë e interfaces.

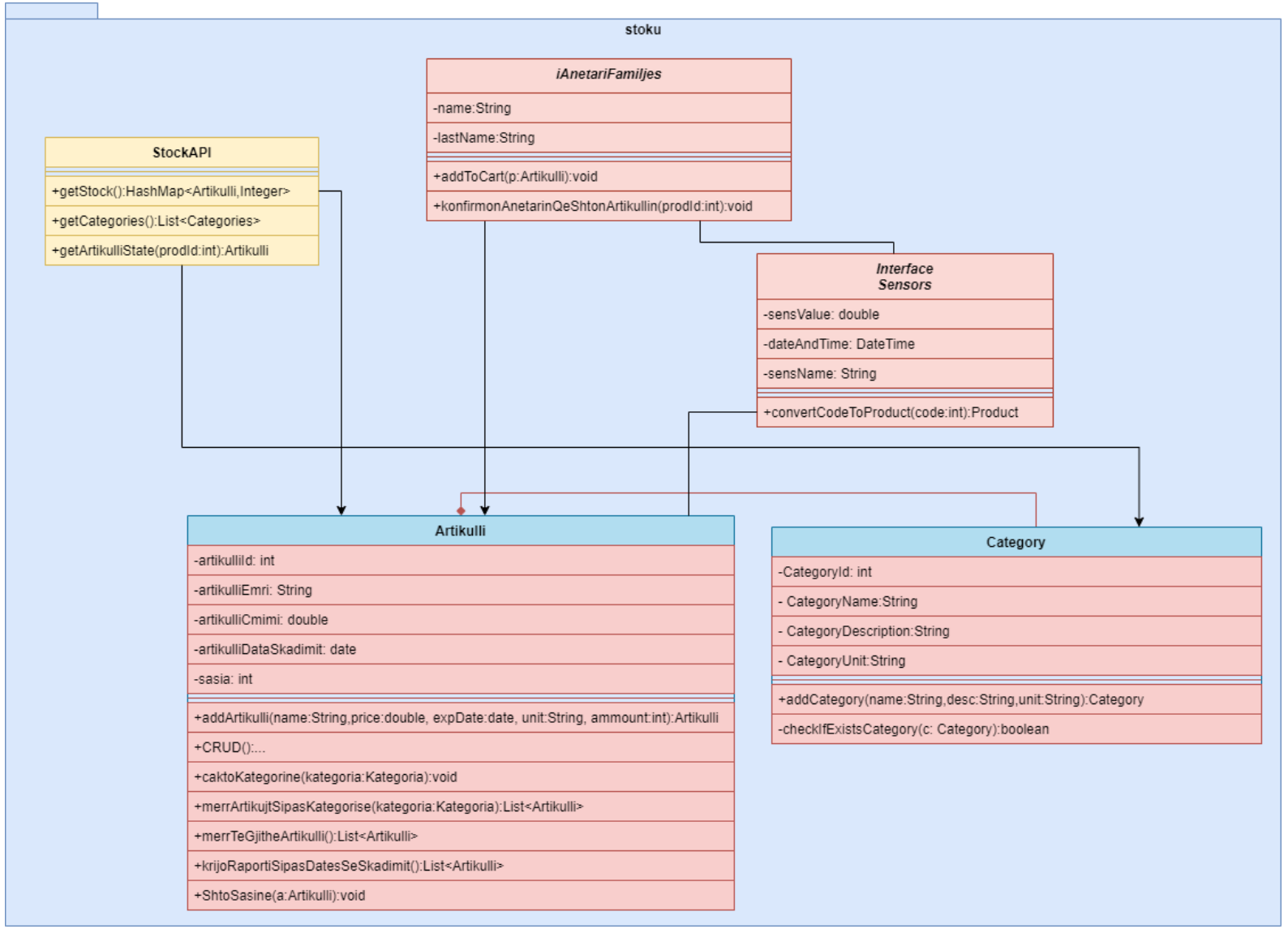
Diagrami i disa moduleve dhe klaseve në aplikacionin SmartFridge



Diagrami i disa moduleve dhe klaseve në aplikacionin SmartFridge….

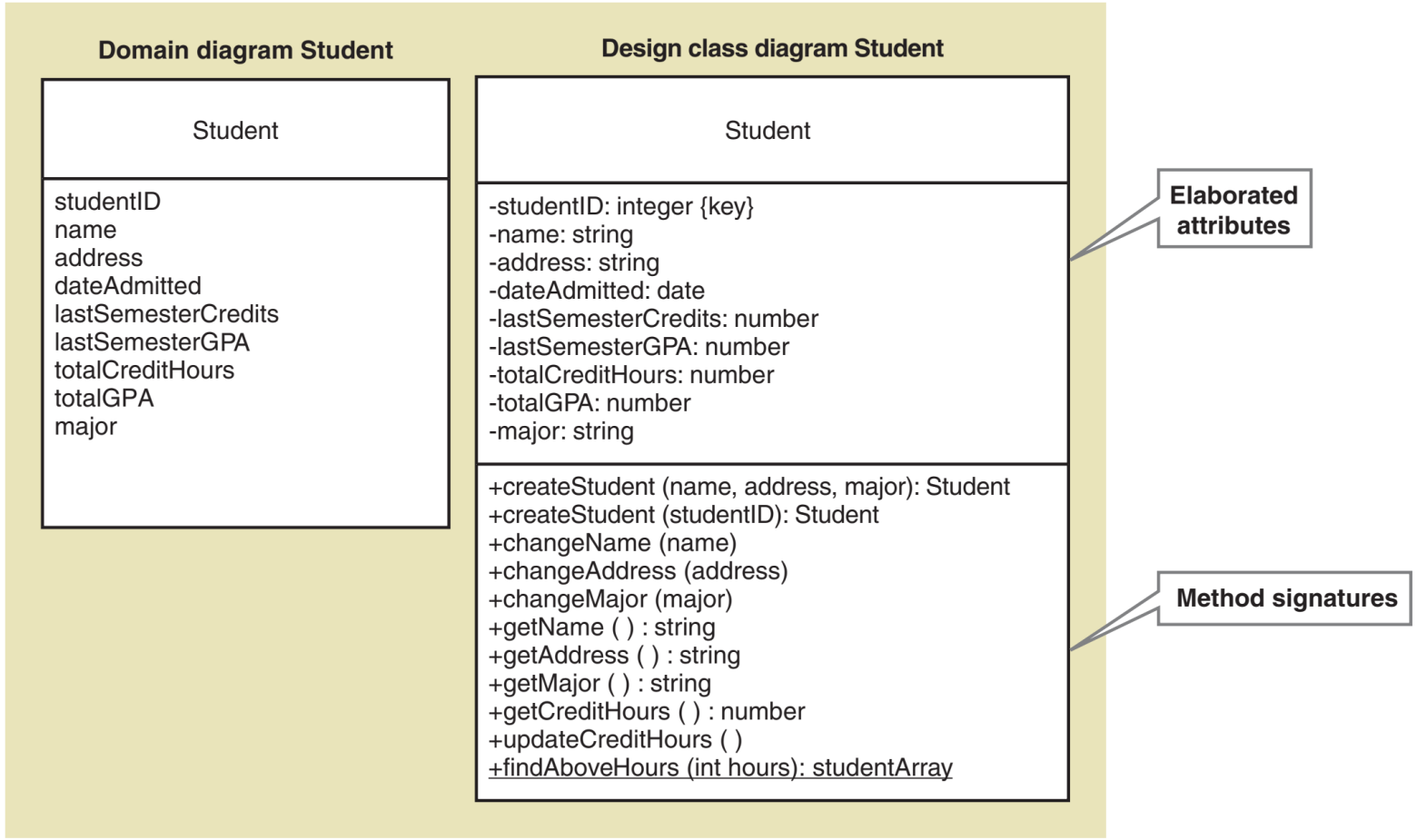






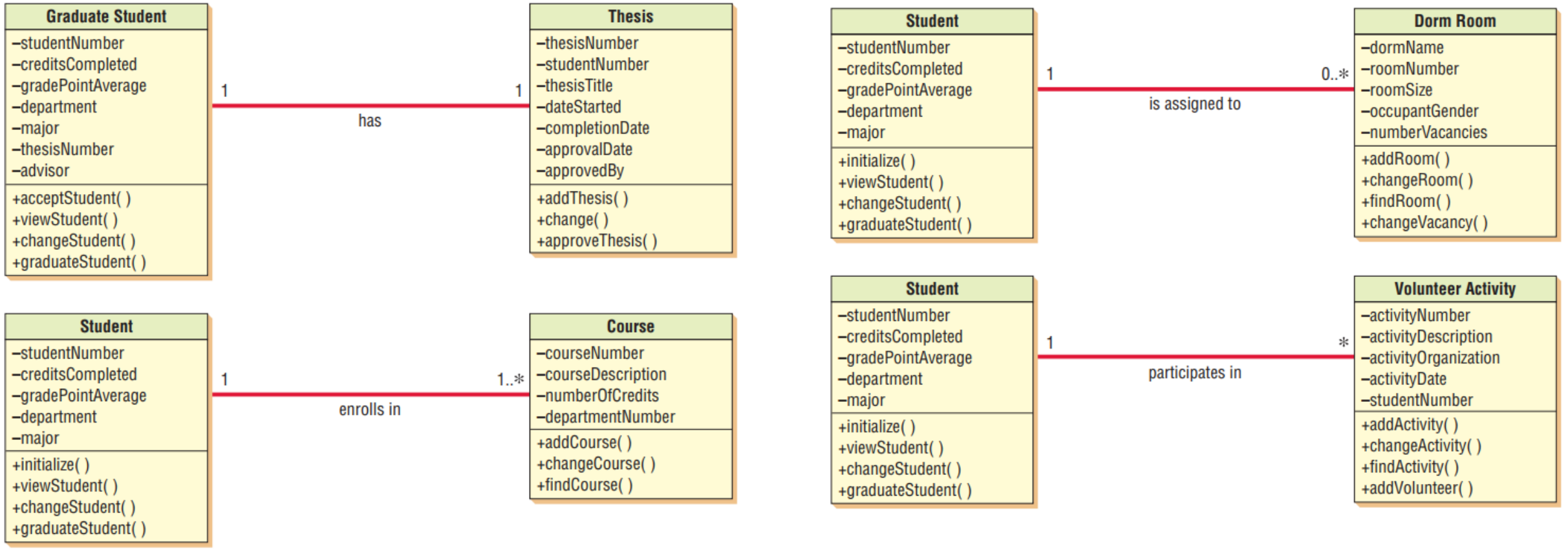
Shembull tjere të diagrameve të klases dhe relacioneve mes tyre

shembuj: të klasës së studentëve
të klasën e ***domenit*** dhe të diagramet i ***dizajnit të klasës***

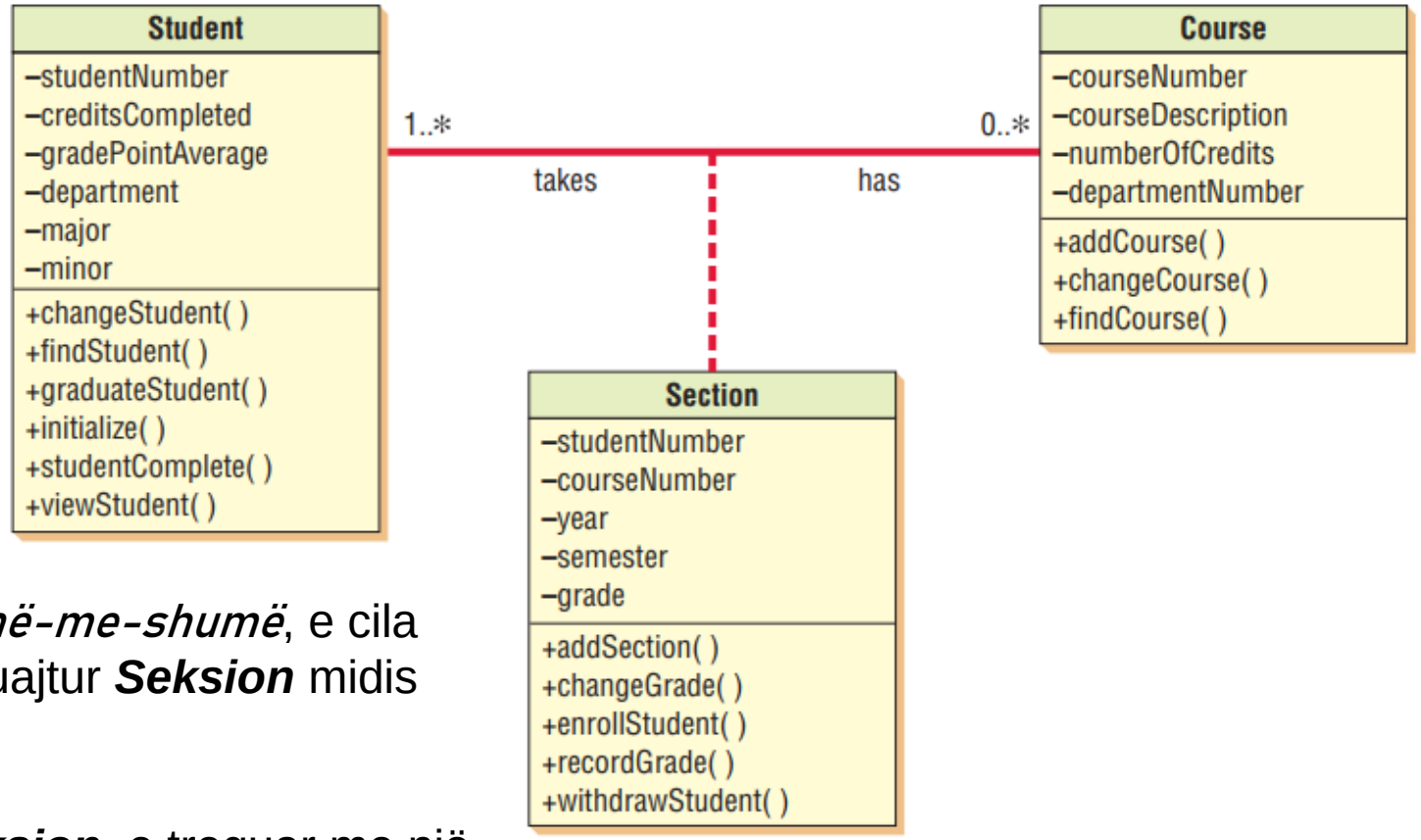



```java
public class Student
{
	//attributes
	private int studentID;
	private String firstName;
	private String lastName;
	private String street;
	private String city;
	private String state;
	private String zipCode;
	private Date dateAdmitted;
	private float numberCredits;
	private String lastActiveSemester;
	private float lastActiveSemesterGPA;
	private float gradePointAverage;
	private String major;

	//constructors
	public Student (String inFirstName, String inLastName, String inStreet,
		String inCity, String inState, String inZip, Date inDate)
	{
		firstName = inFirstName;
		lastName = inLastName;
		...
	}
	public Student (int inStudentID)
	{
		//read database to get values
	}

	//get and set methods
	public String getFullName ( )
	{
		return firstName + " " + lastName;
	}
	public void setFirstName (String inFirstName)
	{
		firstName = inFirstName;
	}
	public float getGPA ( )
	{
		return gradePointAverage;
	}
	//and so on

	//processing methods
	public void updateGPA ( )
	{
		//access course records and update lastActiveSemester and
		//to-date credits and GPA
	}
}
```

llojet e ***asociacionev*** që mund të kemi në ***diagramet e Klasës***



Shembull: një ***klase asociative*** në të cilën një ***seksion*** i veçantë përcakton relacionet midis
Studentit dhe **një Kursi**



Studenti dhe **Kursi** kanë një relacion ***shumë-me-shumë***, e cila zgjidhet duke shtuar një klasë asocuar të quajtur ***Seksion*** midis klasave ***Studenti*** dhe ***Kursi***.

Diagrami ilustron një relacion të quajtur ***Seksion***, e treguar me një *linjë të ndërprer* të lidhur të *shumë-me-shumë* linja relacionesh

Një *Agregacion* përshkruhet shpesh si një relacion "ka një (has a)". Agregacioni ofron një mjet për të treguar se i gjithë objekti është i përbërë nga shuma e pjesëve të tij (objekte të tjera).
Në shembullin e ***regjistrimit të studentëve***, ***departamenti*** ka një ***kurs*** dhe ***kursi*** është për një ***departament***.
Ky relacion është një relacion më e dobët, sepse një ***departament*** mund të ndryshohet ose hiqet dhe ***kursi*** *mund të ekzistojë* ende. Diamanti në fund të linjës së relacioni nuk është plotë (jo i mbushur).

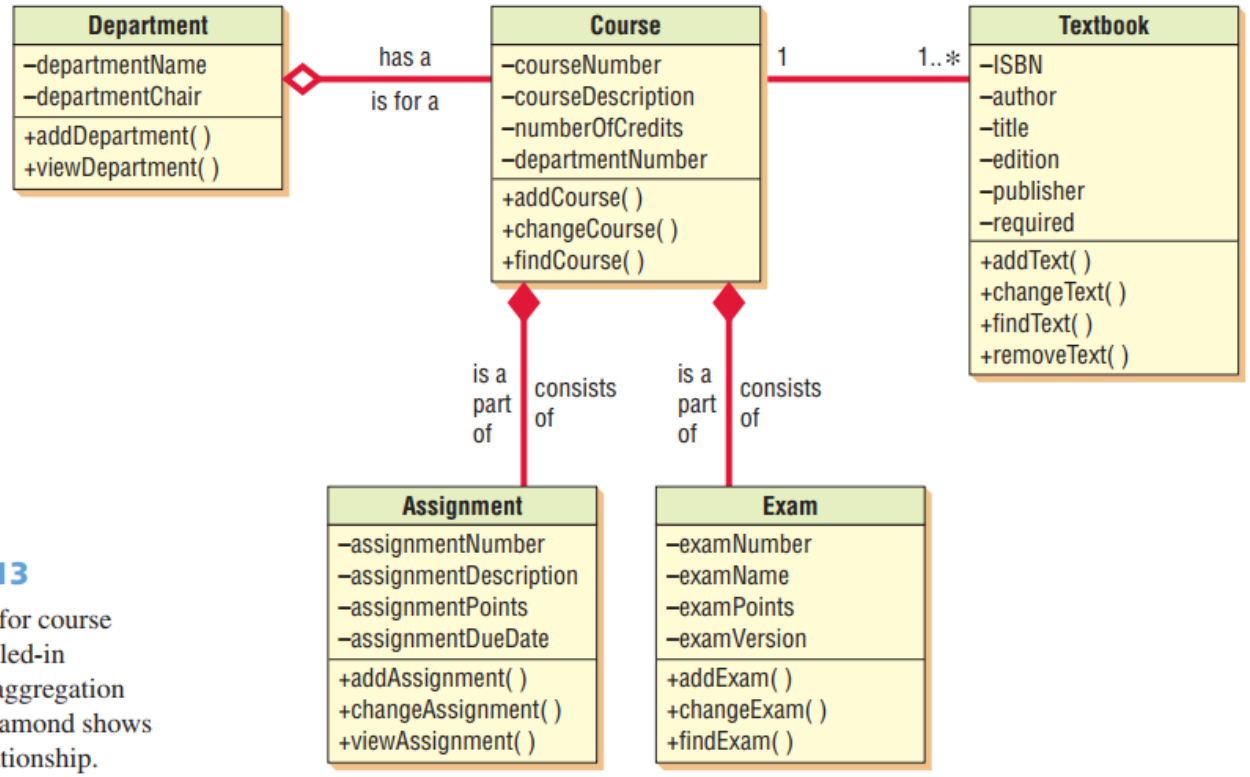
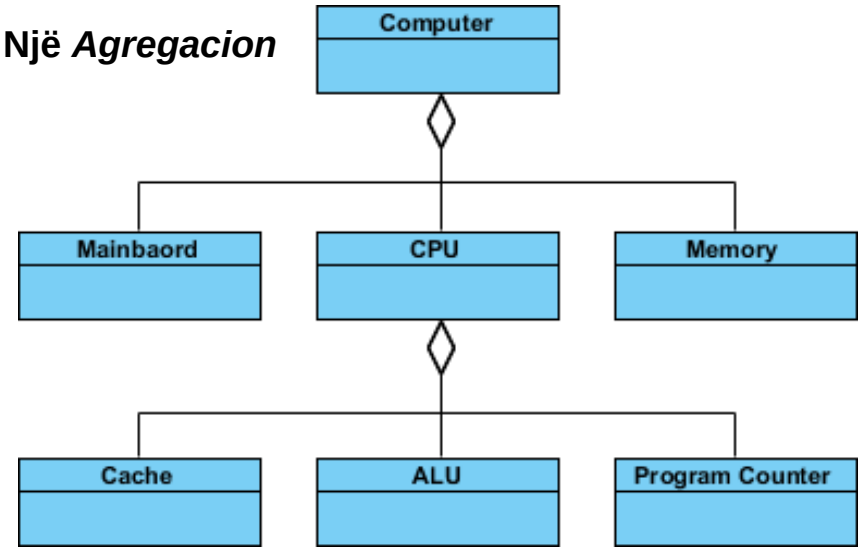


FIGURE 10.13
A class diagram for course offerings. The filled-in diamonds show aggregation and the empty diamond shows a whole-part relationship.

Një *Agregacion*



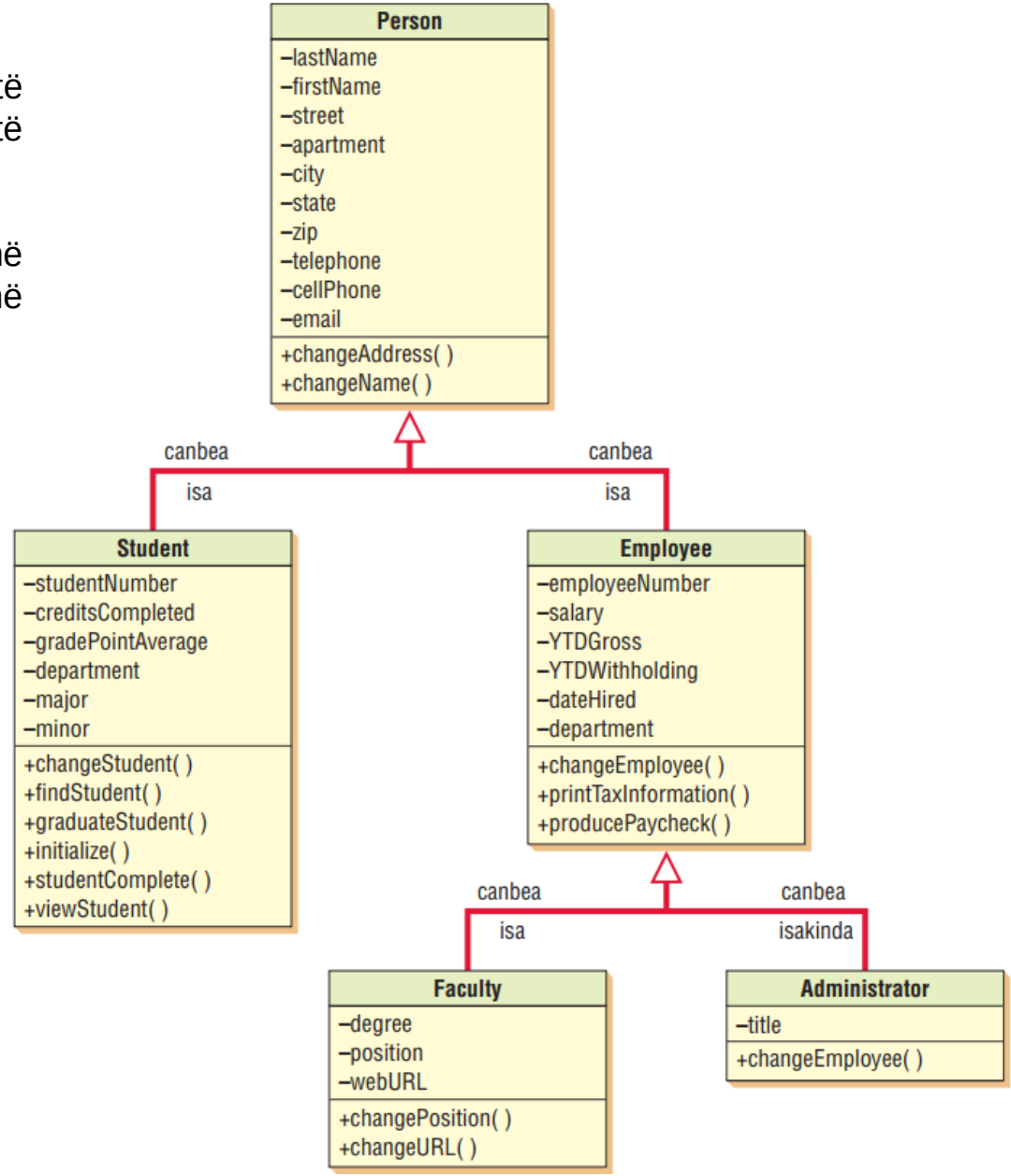
Kompozimi (Përbërja), një relacion *tërësa/pjesës* në të cilën e tërë ka një përgjegjësi për pjesën, është një relacion më e fortë dhe zakonisht tregohet me ***një diamant të mbushur***.
Nëse i tërë është fshirë, të gjitha pjesët fshihen. Për shembull, ekziston një relacion midis ***asigmenti*** (detyre) dhe një ***kursi***, si dhe midis një ***kursi*** dhe një ***provimi***.
Nëse ***kursi*** është *fshirë*, ***asigmenti*** dhe ***provimi*** <u>*fshihen*</u> gjithashtu.

Një ***gjeneralizim*** përshkruan një relacion midis një lloji të ***përgjithshëm*** të gjërave dhe llojeve më ***specifike*** të gjërave. Ky lloj relacioneve është përshkruar shpesh si një relacion "***është një (is a)".***

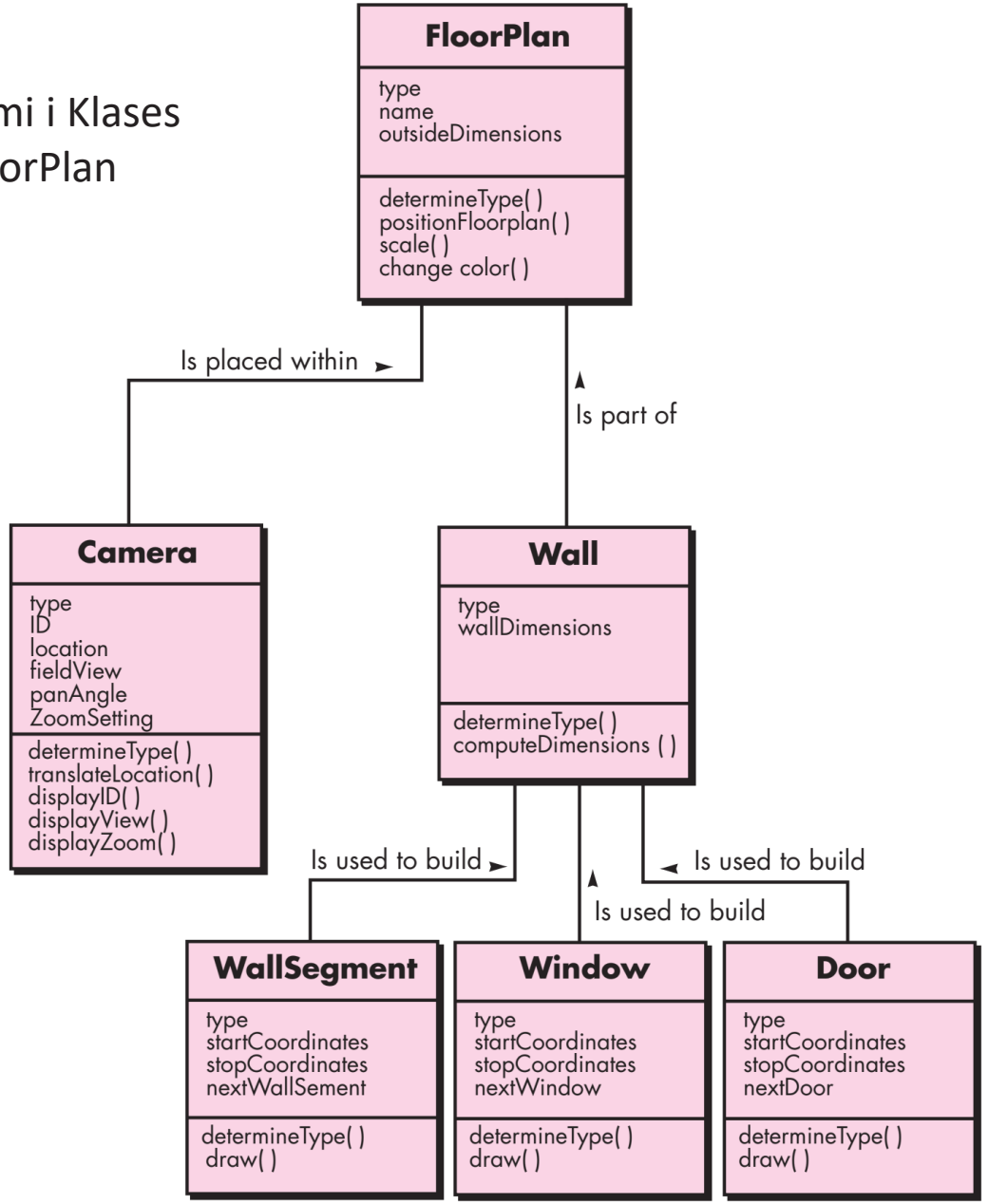
P.sh, një ***makinë*** është një ***automjet*** dhe një ***kamion*** është një ***automjet***, në këtë rast, ***automjeti*** është një gjë gjenerale, ndërsa ***makina*** dhe ***kamioni*** janë gjërat më **specifike**.

Trashëgimia. Disa klasa mund të kenë të njëjtat ***atribute*** dhe/ose ***metoda***. Kur kjo ndodhë, krijohet një *klasë e generale* që përmban *atributet* dhe *metodat* e përbashkta. Klasa e specializuar trashëgon ose merr *atributet* dhe *metodat* e klasës së gjenerale.

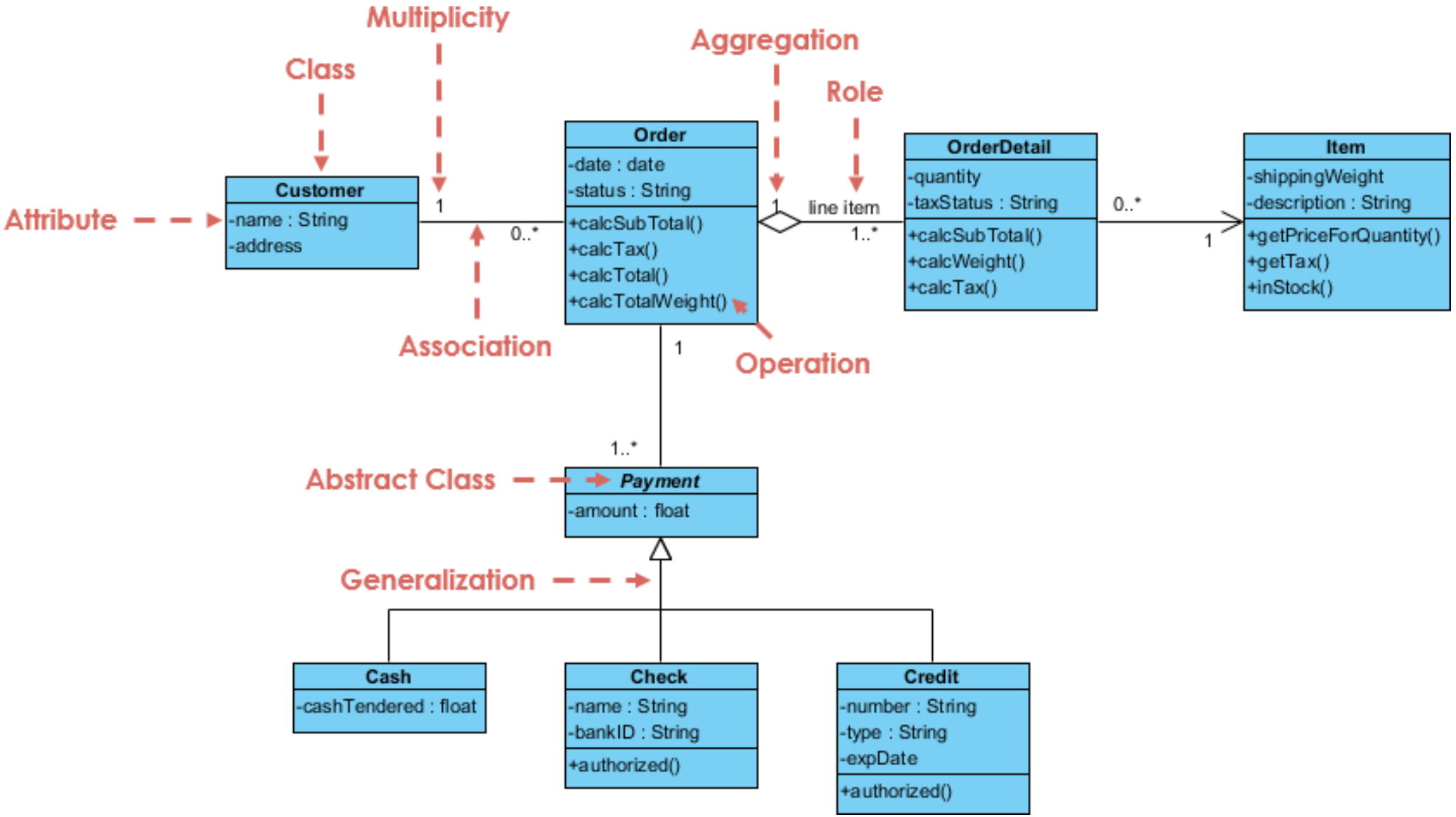
Përveç kësaj, klasa e ***specializuar*** ka ***atribute*** dhe ***metoda*** që janë **unike** dhe të përcaktuara ose definuar ***vetëm*** në *klasën e specializuar*



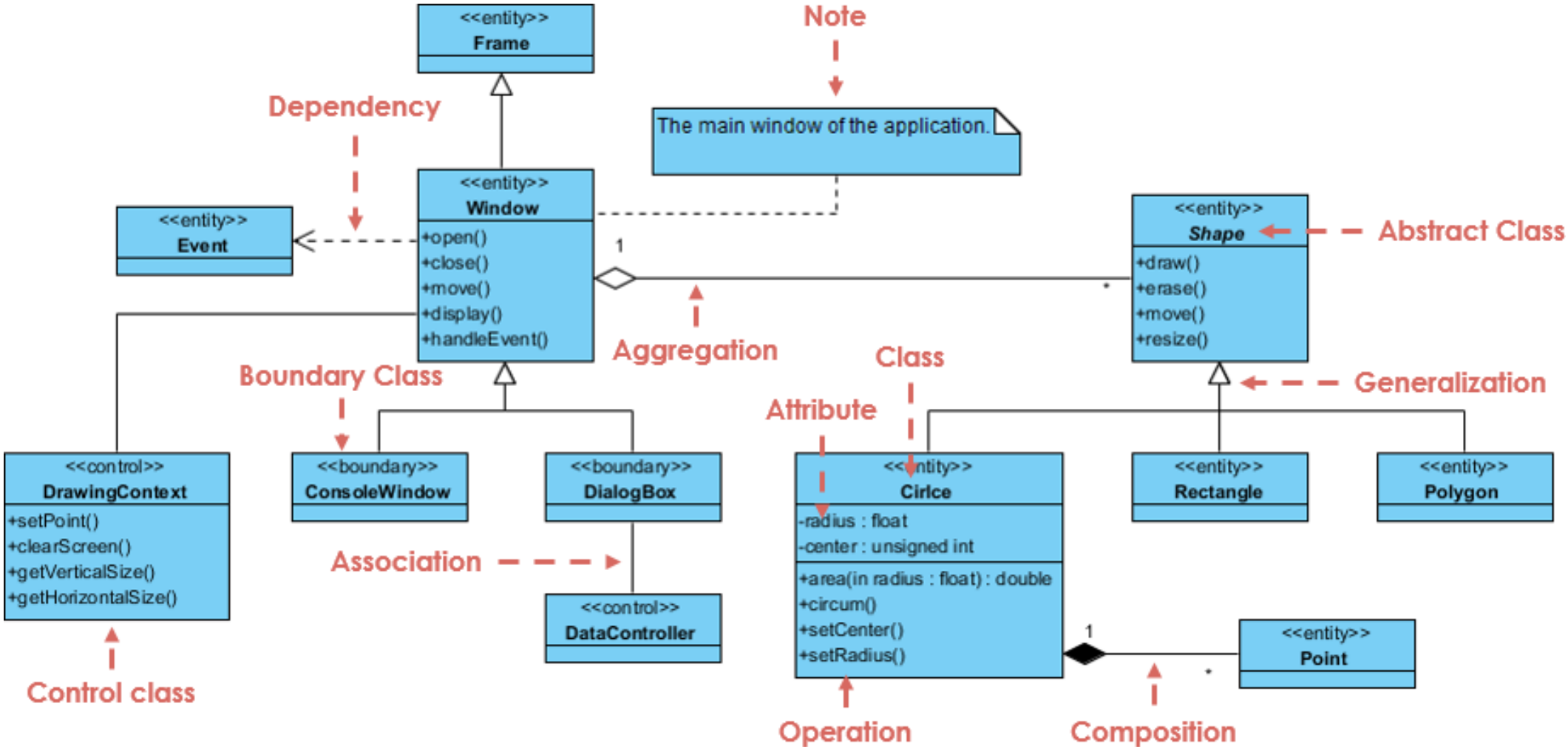
Diagrami i Klases
për FloorPlan

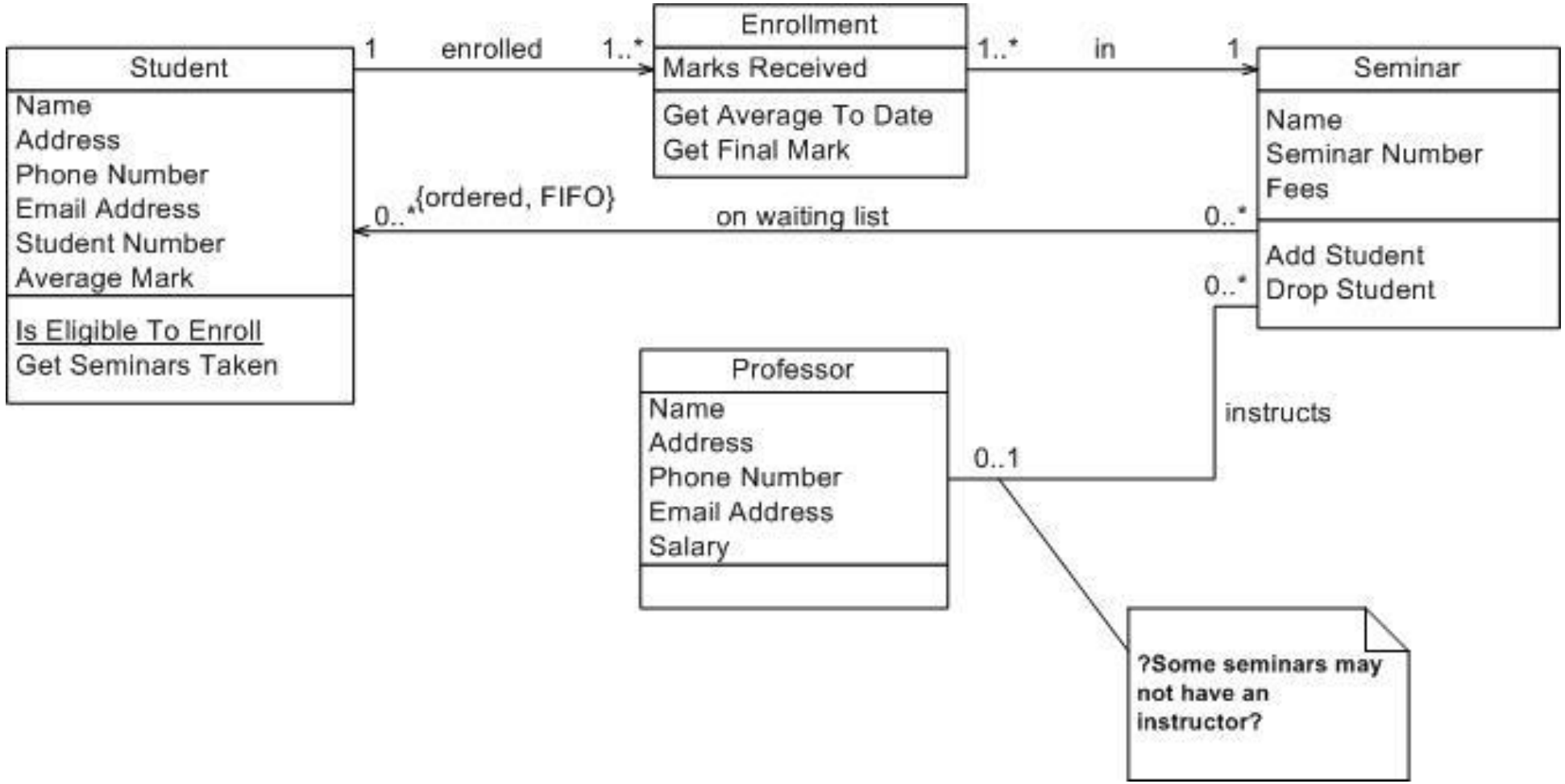


Diagrami i klasës në UML, Rasti i blerjës online



Diagrami i klasës në UML, Rasti i GUI-it




Student
Name
Address
Phone Number
Email Address
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
1
enrolled
1..*
Enrollment
Marks Received
Get Average To Date
Get Final Mark
1..*
in
1
Seminar
Name
Seminar Number
Fees
Add Student
Drop Student
0..*{ordered, FIFO}
on waiting list
0..*
0..*
instructs
Professor
Name
Address
Phone Number
Email Address
Salary
0..1
?Some seminars may not have an instructor?

Kur përdoret diagramet e klasës

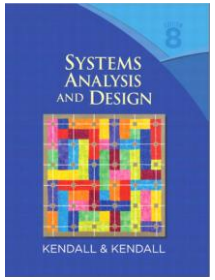
- Sa herë që doni të modelit/dizajnoni për gjërat në një sistem, dhe se si ato lidhen me njëri-tjetrin (i quajtur nganjëherë *modelimit të dhënave*).
- Jep një pamje të përqëndruar në të dhëna të dobishme për:
  - Planifikimi i klasave që nevojiten në OOP
  - Vendimarrje mbi skemën për bazat e të dhënave
- Por keni kujdes që të mos anashkaloni në detaje!
  - E zakonshme për të injoruar disa aspekte
  - P.sh. Modelimi e Klasve dhe asociacionet, por jo veti (atributet) ose operacione

Pyetje

Faleminderit...!

Referencat

- Kapitulli 10: Systems analysis and design 8 Ed. By Kenneth E. Kendall



- Kapitulli 5: Software Engineering. 9th ed. By Ian Sommerville